2011-08-29

Deriving the Cost of Software Maintenance for Software Intensive Systems

Bradley J. Sams

NPS-CE-11-160

ACQUISITION RESEARCH
SPONSORED REPORT SERIES

Deriving the Cost of Software Maintenance for Software Intensive Systems

29 August 2011

by

Major Bradley J. Sams, USMC

Advisors: Dr. John Osmundson, Associate Professor, and

Brad Naegle, Senior Lecturer

Graduate School of Operational & Information Sciences

Naval Postgraduate School

Approved for public release, distribution is unlimited.

Prepared for: Naval Postgraduate School, Monterey, California 93943

The research presented in this report was supported by the Acquisition Chair of the Graduate School of Business & Public Policy at the Naval Postgraduate School.

To request Defense Acquisition Research or to become a research sponsor, please contact:

NPS Acquisition Research Program
Attn: James B. Greene, RADM, USN, (Ret.)
Acquisition Chair
Graduate School of Business and Public Policy
Naval Postgraduate School
555 Dyer Road, Room 332
Monterey, CA 93943-5103
Tel: (831) 656-2092
Fax: (831) 656-2253
E-mail: jbgreene@nps.edu

Copies of the Acquisition Sponsored Research Reports may be printed from our website www.acquisitionresearch.org.

ABSTRACT

Throughout software's lifetime, changes are introduced to the code in order to maintain the desired performance. These changes often create side effects, which cause other cascading effects elsewhere in the software or other system components with which the software interfaces. In a sense, the software degrades because of the maintenance performed on it, not because of a lack of maintenance upkeep. This pattern makes the cost of software maintenance difficult to predict, given the amount of variability in the upkeep process. Therefore, the best that program managers can hope for are heuristics that permit them to approximate annual operating budgets when calculating total ownership costs. Typically, these methods employ metrics used during development to estimate the annual cost of maintaining the software (i.e., source lines of code or function points).

Through correlation and regression analysis, this thesis examines 62 programs that captured software maintenance data to determine a cost model for software maintenance. Even though a model was not built, the main contribution of this thesis is to provide a greater awareness of the complexity of estimating the costs for software maintenance. Additionally, this thesis provides insight to cost variables that may assist program managers when estimating annual software maintenance costs.

THIS PAGE INTENTIONALLY LEFT BLANK

ABOUT THE AUTHORS

Bradley J. Sams, Major, U.S. Marine Corps, Student, Graduate School of Operational and Information Sciences. Maj Sams earned a BA in 1998 and an MS in Education in 2006 from Old Dominion University; he is also a graduate of the U.S. Marine Corps Expeditionary Warfare School and Command and Staff College Distance Education Program. Maj Sams is a field artillery officer, having served in a variety of operational and staff assignments in 13 years of active service, including his most recent tour with 1st Battalion, 12th Marines. Upon graduation from the Information Systems and Technology program at the Naval Postgraduate School in September 2011, Maj Sams will report to Marine Corps Systems Command, Quantico, VA.

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

There are many people who have helped this thesis become a reality. First, Brad Naegle, LTC, USA (Ret.); his encouragement and unmitigated confidence in my abilities have been an enormous inspiration. This thesis never would have gotten off the ground without the determination and patience of Greg Mislick, LtCol, USMC (Ret.) in the Operations Research Department. His assistance in obtaining data for this thesis was priceless.

Thanks to Dr. John Osmundson for listening to me prattle on about software engineering program management and giving me a gentle nudge in the right direction when needed. A tremendous amount of thanks to Dr. Wilson Rosa and Peter Braxton as well as the professionals at the Air Force Cost Analysis Agency and Technomics for providing their expertise and advice during my data collection travels. Their support was unequivocal and greatly appreciated. Thanks also go out to Mike Popp in the NAVAIR 4.2 Cost Department for taking time away from his busy schedule to listen to a struggling artilleryman ask questions about software cost estimation.

Additionally, I would like to thank the Acquisition Research Program, especially RADM James Greene, USN (Ret.), Ms. Karey Shaffer, and Ms. Tera Yoder, for providing funding and resources to ensure the success of this thesis.

I'd like to thank Audrey, Catherine, and Nikolaus for giving up their time and attention with me so that I could complete this thesis. Even though I can never relive my time away from them, I hope that in the future they can appreciate that sometimes you have to make hard choices in order to accomplish good work. My final recognition goes to my wife, Crystal, who was subjected to several hours of unending babble, but who made me feel like my topic was the most important subject in academia. I cannot ever hope to match her limitless patience, her enduring support, and her boundless encouragement. I would be a lesser man without her. Thank you, Kiddo.

THIS PAGE INTENTIONALLY LEFT BLANK

NPS-CE-11-160

ACQUISITION RESEARCH SPONSORED REPORT SERIES

Deriving the Cost of Software Maintenance for Software Intensive Systems

29 August 2011

by

Major Bradley J. Sams, USMC

Advisors: Dr. John Osmundson, Associate Professor, and

Brad Naegle, Senior Lecturer

Graduate School of Operational & Information Sciences

Naval Postgraduate School

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

ACE	Air Combat Equipment
AFCAA	Air Force Cost Analysis Agency
AMC&D	Advanced Mission Computer and Display
ASE	Aviation Support Equipment
CAINS	Carrier Aircraft Inertial Navigation System
CARC	Chemical Agent Resistant Coating
CASS	Consolidated Automated Support System
CDP	Capability Defect Package
CMM	Capability Maturity Model
CMMI	Capability Maturity Model–Integrated
COCOMO	Constructive Cost Model
COTS	Commercial Off-The-Shelf
CSCI	Computer Software Configuration Items
CSFIR	Crash Survivable Flight Incident Recorder
DCAPE	Defense Cost Assessment and Program Evaluation
DCARC	Defense Cost and Resource Center
DID	Data Item Description
EWSSA	Electronic Warfare Software Support Activity
FRA	Fleet Response Activity
FWA	Fixed Wing Aviation
FY	Fiscal Year
GPS/CDNU	Global Positioning System/Control Display Navigation Unit
GPSW	Ground Warning Proximity System
IPT	Integrated Product Team
ISPAN	Integrated Strategic Planning and Analysis Program

MDAP	Major Defense Acquisition Program
MIS	Missiles
NAVAIR	Naval Air Systems Command
PRE	Program Related Engineering
ODASA–CE	Office of the Deputy Secretary of the Army for Cost and Economics
PDSS	Post Deployment Software Support
QSM	Quantitative Software Management
ROI	Return on Investment
RWA	Rotary Wing Aviation
SEER–IT	System Evaluation and Estimation of Resources–Information Technology
SEER-SEM	System Evaluation and Estimation of Resources–Software Engineering Model
SIS	Software Intensive System
SLIM	Software Lifecycle Management
SLOC	Source Lines of Code
SRDS	Structural Data Recording Set
SRDR	Software Resources Data Report
SSA	Software Support Activity
TAMMAC	Tactical Aircraft Moving Map Capability
TAWS	Terrain Awareness Warning System
TOC	Total Ownership Cost
VAMOSC	Visibility and Management of Operating and Support Costs
WRALC	Warner Robins Air Logistics Center

EXECUTIVE SUMMARY

Software is becoming frequently more ubiquitous in the systems the Department of Defense procures. These systems are increasingly reliant on software to successfully perform their missions. This reality places greater emphasis on ensuring the accompanying or embedded software performs as expected. However, reliability is not cheap and trends toward a greater proportion of the system sustainment cost. In an age of rapidly decreasing funds to support government functions (including the military), total ownership cost has garnered a great deal more attention than in previous system procurement. Previous studies have shown the disproportionate annual cost of maintenance as compared to the software's development, and program managers require accurate models in order to estimate the life-cycle costs for proposed systems. Many models exist to provide estimates for software development cost, but few are able to predict the cost to support software once delivered to the end user.

The researcher examined over 60 programs that captured software maintenance data. Given the diverse nature of the data set provided, the cost to support software was analyzed from different perspectives. The research calculated correlations and performed regressions on the data to derive the most promising relationships and candidate models that might reveal some insight into the influence of particular variables related to cost.

The observations of these results revealed that a reliable and consistent model could not be created from the data provided. However, it was determined from this limited data set that source lines of code were not an adequate predictor of maintenance cost. The number of defects reported divulged the strongest relationships with regard to influencing cost. Additionally, the number of computer system configuration items could provide a useable factor when estimating the cost of maintenance. Lastly, the researcher recommends a uniform means for software support agencies or contractors to report their software maintenance efforts, similar to the mandated software resources data report.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

> The total cost of maintaining a widely used program is typically 40 percent or more of the cost of developing it. Surprisingly, this cost is strongly affected by the number of users. More users find more bugs.
>
> – Frederick P. Brooks, Jr. (1995)

A. BACKGROUND

The current trend in government spending and appropriation is austerity. As U.S. commitments in Iraq draw to a close and as efforts in Afghanistan are tailored to a smaller force, the U.S.'s attention will be increasingly focused on reducing the budget deficit and strengthening the domestic economy. Secretary of Defense Robert Gates declared that "the gusher is off" ("Defense Spending," 2010), referring to the last several decades of increasing defense budgets. Since the Department of Defense (DoD) accounts for over 50% of discretionary funding by the government, the concern for how the military spends its funds will garner more interest and be a target for closer scrutiny. Recent acquisition policy directives aimed at capturing the total ownership cost (TOC) underscore this reality. For example, the Weapons Systems Acquisition Reform Act (WSARA, 2009) instructs the Defense Cost Assessment and Program Evaluation (DCAPE) to review assessment methods for operations and support costs for major defense acquisition programs (MDAP). Additionally, the accompanying DoD Directive Type Memorandum (DTM) 09-027 charges the Milestone Decision Authority (MDA) to competitively contract for the maintenance and support contracts for its programs (Under Secretary of Defense for Acquisition, Technology, and Logistics [USD(AT&L)], 2009). The increased emphasis on operations and support costs challenges acquisition professionals to ensure that the programs they acquire are sustainable in future years by a decreasing operations budget.

Software maintenance implies the ability to make corrections, change functionality, or perfect previously identified flaws in the functionality of the software. These actions are typically executed during the operations and support phase of the acquisition life cycle. Maintenance on software is very different from that completed on hardware. For example, it is easy to observe early on when a piece of hardware needs attention. There are clear warning signs given to the operator well before the piece of equipment breaks and ceases to function (rust on joints, leaks at welds, etc.). However, software does not provide these signals; it slows to unacceptable performance levels, freezes, hangs up, or simply stops functioning without warning and leaves the operator without the ability to execute the mission. Estimating the cost of maintaining hardware can be done easily by simply following the manufacturer's guidance on preventative maintenance before the problem becomes corrective in nature. The cost associated with this maintenance can then be extrapolated across the expected life of the hardware in order to derive a number to justify budgets. Software is inherently complex and, therefore, more difficult to accurately estimate the maintenance effort required to support it. During the development of the software, program managers (and, ultimately, the maintainers) are not able to accurately predict when the software is going to need to be upgraded or perfected or when it might crash unexpectedly. Therefore, the best that program managers can hope for are heuristics that permit them to approximate annual operating budgets. Typically, these methods employ metrics used during development to estimate the annual cost of maintaining the software (i.e., source lines of code or function points). In his article, Sneed (2004) commented on the imprecision of predicting development costs to estimate maintenance costs. This situation presents a dilemma when the heuristics that program managers rely upon are based on erroneous assumptions and imprecisely calibrated cost factors.

B. PURPOSE

The purpose of this thesis is to present an analysis of several cost-related factors involved in software maintenance and their influence across different application domains. This information could then be used by program managers to derive a cost-estimation relationship and, ultimately, a cost model to determine the forecasted annual cost to support similar systems while still in development. It is the researcher's belief that such a software maintenance cost model would more accurately portray the total ownership cost of a particular system than current methods.

C. RESEARCH QUESTIONS

1) What cost factors are involved when a program manager estimates the post-deployment software support (PDSS) for a software-intensive system (SIS)?

2) Is there a model that can be derived for program managers to use in order to more accurately estimate the total life-cycle (or operational) cost of software-intensive command and control or weapons systems?

3) Is there a better method for program managers to budget software maintenance rather than comparing the development costs to anticipated post-deployment support?

4) What software maintenance information is necessary in order to derive a reliable cost model for program managers?

D. BENEFITS OF THE STUDY

This thesis presents an analysis of different factors related to the cost of existing software intensive systems from a variety of domains. This information can be employed by acquisition managers during the development phase of the acquisition life cycle to predict the costs associated with the software maintenance support for a similar system. This data could then be used to calibrate existing heuristics and more accurately estimate the TOC for a proposed system.

E. SCOPE

This thesis is limited to the factors provided by the Naval Air Systems Command (NAVAIR) and the various programs participating in the Air Force Cost Analysis Agency (AFCAA) software maintenance study. While there are an indefinite amount of factors that contribute to the cost of software maintenance, this thesis only analyzes those categories collected in order to derive correlation coefficients and candidate cost-estimating relationships through regression.

F. METHODOLOGY

This thesis used three analysis methods. First, several literature sources related to software maintenance were examined. Additionally, three of the most popular software cost-estimation techniques were researched to understand how these methods estimate post-deployment software support. Second, the data collected from the various sources was presented and described. Third, the data collected was analyzed for any correlations or cost-estimating relationships that could be derived and employed in an appropriate model for post-production software support. Lastly, results of the data analysis presented recommendations for program managers concerned with the total operational costs of proposed software intensive systems.

G. ORGANIZATION OF THESIS

In Chapter II, the researcher provides relevant definitions for software maintenance from an assortment of sources. Additionally, techniques for estimating software maintenance are presented from three prevalent cost models used by professionals.

In Chapter III, the researcher describes the data collected from NAVAIR and the AFCAA study on software maintenance. This chapter depicts the disparate categories of data that are analyzed in the following chapter.

In Chapter IV, the researcher analyzes the data presented in Chapter III through the conduct of bivariate correlations and simple linear regressions. The results of this analysis then determines the strongest cost-estimating relationships based on the limited amount of data available.

In Chapter V, the researcher presents the conclusions of this analysis and makes recommendations to program managers for estimating the cost of post-deployment software support based on the categories analyzed in Chapter III. This chapter also makes recommendations for further research on the software maintenance topic.

II. SOFTWARE MAINTENANCE AND COST-ESTIMATION MODELS

> Software maintenance is usually, explicitly or not, the largest single element of developing, owning, and operating a software system.
>
> – Christensen and Thayer (2001)

A. SOFTWARE MAINTENANCE

Software does not possess the same physical characteristics as hardware. End users cannot scrub the rust off existing software, apply a coat of chemical agent resistant coating (CARC) and make it look as good as new. In fact, end users may not even be able to see that their software possesses rust at all. However, software does degrade. Throughout software's lifetime, changes are introduced due to poor quality development or other situations that mandate software alterations. These changes often create side effects that are incorporated into the software, which causes cascading effects elsewhere in the software or in other system components with which the software interfaces. In a sense, the software degrades because of the maintenance performed on it, not because of a lack of maintenance upkeep. Additionally, software maintenance does not permit the notion of spares. For example, when a truck's serpentine belt is broken, a suitable replacement belt can be changed out for the defective one. This example does not correlate well to software, as the truck's architecture is not altered by the belt replacement, but software maintenance typically does alter the software architecture. A maintainer is unable to simply replace the degraded piece of software with a fresh one. In order to avoid the unintended consequence of creating more problems by replacing the defective software, the maintainer would need to redesign the entire software component in order to fix the one particular problem, without creating other problems. Since this resolution is not realistic, patches (frequently referred to as maintenance) are injected in the software to correct deficiencies. These repairs are intended to increase the software's reliability over time. In theory, software should be able to perform as developed throughout its life cycle without issue. Unfortunately, reality is much more complicated.

As demonstrated in Figure 1, the software reliability curve significantly differs from the hardware curve.

Many factors influence the maintenance performed on software, including the repair of defects incorporated in the software during development or because of changes in requirements or the desire to improve performance (Department of the Air Force, 2000). These aspects shape the reliability curve differently than anticipated for software. As mentioned, even these remedies may inadvertently produce greater degradation of the software, which requires more maintenance and the possibility of injecting new defects. This pattern makes the cost of software maintenance difficult to predict, given the amount of variability in the maintenance process. These are the environmental circumstances in which the program manager, the developers, and the maintainers find themselves when creating a realistic annual cost estimate as the software ages.

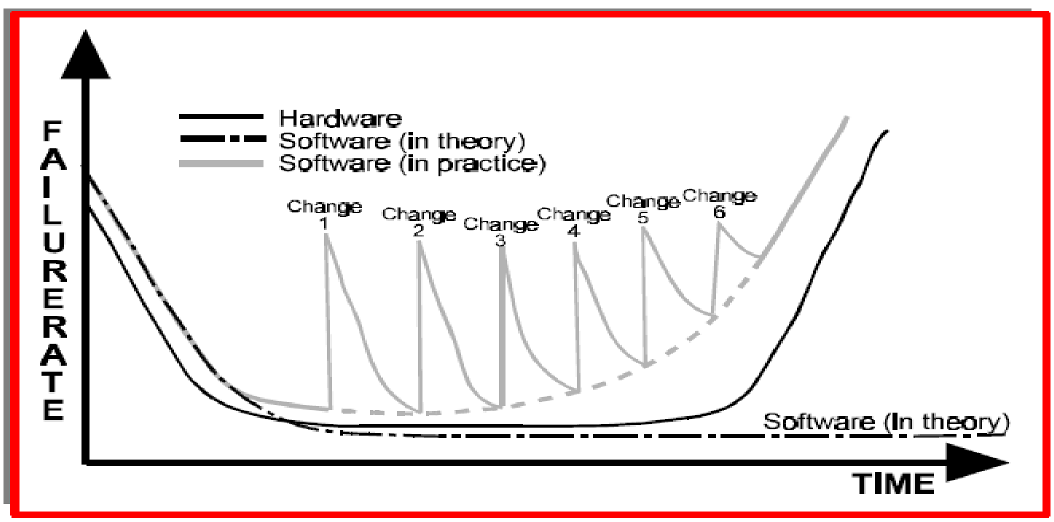


Figure 1. Bathtub Curves for Hardware and Software
(Department of the Air Force, 2000)

In order to adequately discuss this topic, it is important to provide an operational definition for software maintenance that can be used throughout this thesis. There have been a wide variety of opinions on what constitutes software maintenance, as shown in Table 1. It is no surprise that in this chronological list of generally accepted definitions for software maintenance each definition mentions that support occurs after its delivery. Additionally, these definitions refer to software changes or modifications, but only the most recent description mentions the cost associated with software support. It is the associated cost of maintenance that will occupy the attention of program managers.

Table 1. Overview of the Often-Quoted Definitions of Software Maintenance
(Abran & April, 2008)

Definition	Year
"Changes that are done to software after its delivery to the user."	1983
"The totality of the activities required in order to keep the software in operational state following its delivery."	1984
"Maintenance covers the software life-cycle starting from its implementation until its retirement."	1990
"…modification to code and associated documentation due to a problem or the need for improvement. The objective is to modify the existing software product while preserving its integrity."	1995
"…the modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a modified environment."	1998
"…the totality of activities required to support, at the lowest cost, the software. Some activities start during its initial development but most activities are those following its delivery."	2005

When program managers analyze costs for maintenance, they first need to understand the kind of anticipated maintenance that will represent the majority of support costs. This analysis will influence the scope of the cost estimation and contribute to a better understanding of the effort employed. Nevertheless, the maintenance effort is not limited to making changes only in the source code. As noted by Parthasarathy (2007), maintenance costs include operations and online support, fixing bugs, and enhancing the application (both major and minor changes), which contributes to the total ownership cost of software. However, this thesis limits the definition of software maintenance to three areas shown in Figure 2: corrective, perfective, and adaptive. These groupings exist solely based on the maintenance change expected to be performed.

Adaptive change occurs when the developed software needs to be changed based on external realities. "Classic examples are adapting to an updated operating system, changed or new hardware, software tools, and data format changes" (Christensen & Thayer, 2001, p. 150). Approximately 20% of software maintenance falls in this category (Christensen & Thayer, 2001). Corrective change occurs when the software incurs unanticipated defects. These adjustments can be completed in the course of normal business or take the form of emergency maintenance that needs to be accomplished immediately. Around 20% of software maintenance is corrective in nature (Rendon & Snider, 2008). Lastly, those actions that attempt to improve the software's performance are referred to as perfective maintenance. Similar to corrective, perfective alterations can be planned in conjunction with other work (Christensen & Thayer, 2001). Perfective modifications absorb the remaining 60% of software maintenance. Knowing the types of maintenance and their influence on total effort allows program managers to better analyze costs.

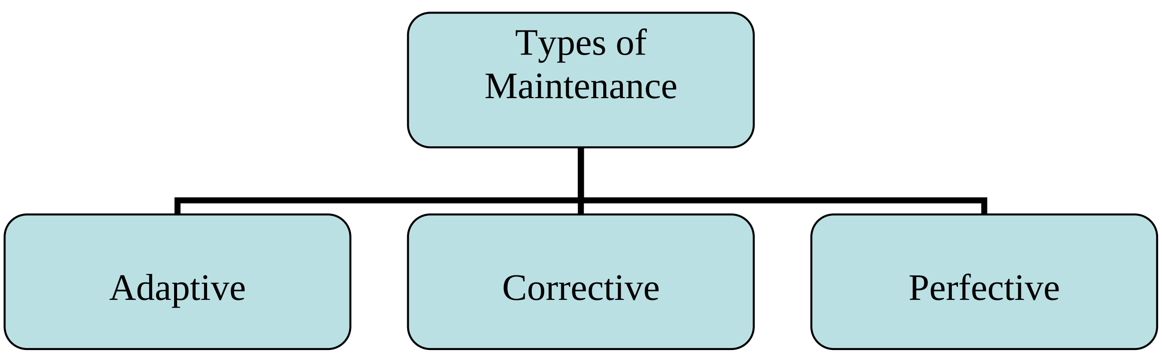


Figure 2. Types of Software Maintenance
(Christensen & Thayer, 2001)

It is accepted that the total ownership cost of software includes the associated cost of maintaining the software beyond development and delivery. However, there are few models that provide program managers the ability to estimate or predict how much it will cost per future year to maintain a particular software project. Therefore, it is rational that practitioners would turn to easily captured development variables as a basis for their post-deployment support costing approximation. For example, Deutsche Post (a mail service in Germany) estimated the maintenance of a new application as a percentage of the development costs (Buchmann, Frischbier, & Putz, 2011). This approach to maintenance estimation was also challenged by Sneed (2004), who said that development costs may not be related to the cost of maintaining a system. In fact, Sneed commented that maintaining a commercial off-the-shelf (COTS) system could cost 40% more than a system created from scratch and that the development of low-priced agile projects were liable to cost more to maintain (Sneed, 2004). Therefore, it is important for program managers to understand the efficacy of their chosen software maintenance cost model, and program managers should appreciate the complexity and challenges connected to sustaining software.

B. COST-ESTIMATION TECHNIQUES

1. Purpose

There are a variety of cost models in existence to estimate the development costs for a software project. Typically, these models consider post-deployment software support as another phase of development. There are very few cost models that exclusively attempt to estimate maintenance cost for software. This section describes three popular cost models that program managers use to estimate maintenance effort, which can be used to approximate costs.

2. Constructive Cost Model II

Developed in 2000, the Constructive Cost Model (COCOMO) II expands Barry Boehm's original software cost-estimation model, COCOMO, written in 1981. COCOMO II continues the principles described in Boehm's earlier work and analyzes "major product rebuilds changing over 50 percent of the existing software, and development of sizable (over 20 percent changed) interfacing systems requiring little rework of the existing system" (Boehm et al., 2000, p. 28). Boehm et al.'s (2000) updated work considers software maintenance through two sections, sizing software maintenance and maintenance effort. Both of these sections assume that "maintenance cost generally has the same cost driver attributes as software development costs" (p. 58). These portions of the COCOMO II method can be used to create an estimate for the size of the maintenance required using the known base code size.

a. Sizing Software Maintenance

A COCOMO II sizing software maintenance model begins by examining the software understanding (SU) of the existing software (determined on a scale from 0–50%), dividing by 100, and multiplying this quotient by the programmer unfamiliarity (UNFM) factors shown in Table 2. The product of these two factors is then added to 1, which produces the maintenance adjustment factor (MAF).

Table 2. Rating Scale for Programmer Unfamiliarity (UNFM)
(Boehm et al., 2000)

UNFM Increment	Level of Unfamiliarity
0.0	Completely Familiar
0.2	Mostly Familiar
0.4	Somewhat Familiar
0.6	Considerably Familiar
0.8	Mostly Unfamiliar
1.0	Completely Unfamiliar

The next portion of the software maintenance size equation comes from the maintenance change factor (MCF). This number can be obtained by placing the sum of modified and added size in the numerator and the known base code size in the denominator, as indicated in Equation 1 from Boehm et al. (2000).

$$\text{MCF} = \frac{SizeAdded + SizeModified}{BaseCodeSize} \qquad (1)$$

Using the MAF and the MCF, the basic equation for the maintenance size can be found in Equation 2, taken from Boehm et al. (2000).

$$(\text{Size})\text{M} = [(\text{Base Code Size}) \times \text{MCF}] \times \text{MAF} \tag{2}$$

b. *Software maintenance effort*

Program managers need to capture the effort required to maintain any existing software in order to justify budget requests and appropriately assign maintenance responsibilities. COCOMO II provides a formula to derive the maintenance effort in person-months (typically 152 hours per month). The estimation formula for maintenance effort can stem from Equation 3 from Boehm et al. (2000).

$$PM_M = A \times (Size_M)^E \times \sum_{i=1}^{15} EM_i \tag{3}$$

where PM_M = person-months effort for maintenance;

A = the effort coefficient that can be calibrated, currently set to 2.94;

$(Size_M)^E$ = the maintenance size with the exponent *E* derived from an aggregation of five scale factors associated with economies of scale (i.e., precedentedness "PREC" and development flexibility "FLEX"; and

EM_i = 15 effort multipliers (minus the required development schedule "SCED" and required reusability "RUSE").

Once PM_M has been derived from Equation 3, the results can be taken further to estimate the average maintenance staffing level (FSPM) associated with the duration of any maintenance activity (TM), as demonstrated in Equation 4 from Boehm et al. (2000).

$$FSPM = PM_M \div TM \tag{4}$$

The ability of a program manager to estimate the number of person-months needed to maintain a certain amount of software could be extremely useful, especially for new software builds without historical analogous systems. COCOMO and COCOMO II are popular methods to determine software cost estimation due to their ubiquity and the lack of cost to the user. However, there are commercial estimation methods that provide program managers the ability to project post-deployment support for a proposed software development.

3. System Evaluation and Estimation of Resources (SEER) Family of Products

Produced by Galorath Incorporated, the System Evaluation and Estimation of Resources (SEER) family of products uses parametric-based models, specifically designed algorithms, a historical database of previous project cost estimations, and sophisticated simulation/modeling engines that produce reports (including a report for maintenance effort by year) based on user inputs and desires. The result is a variety of reports that allow managers and developers to estimate their costs, as displayed in Figure 3.

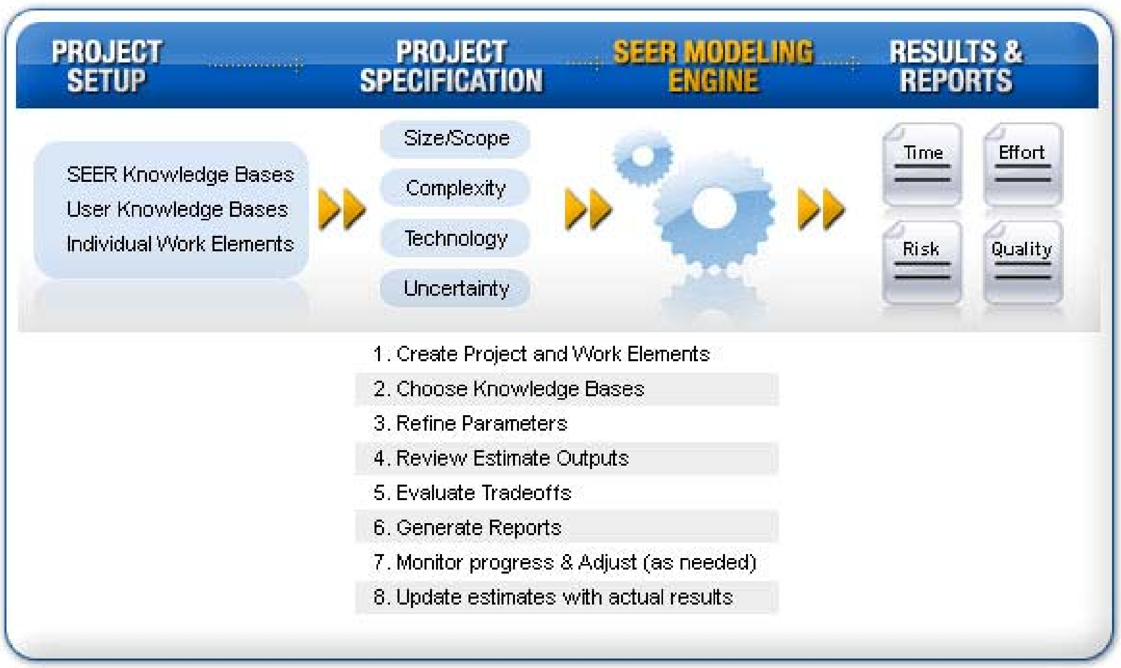


Figure 3. SEER Parametric Modeling Process
(Galorath Incorporated, 2011b)

Two such products from the SEER family are SEER–Software Estimating Model (SEER–SEM) and SEER for Information Technology (SEER–IT). These tools permit managers and developers to estimate the costs associated with software builds. One of the features of these tools includes the ability to estimate the cost of post-deployment support. As depicted in Figure 4, Galorath defines the costs associated with software maintenance by using the following terms and definitions:

- Corrective maintenance—The costs due to modifying software to correct issues discovered after initial deployment (generally 20% of software maintenance costs).
- Adaptive maintenance—The costs due to modifying a software solution to allow it to remain effective in a changing business environment (25% of software maintenance costs).
- Perfective maintenance—The costs due to improving or enhancing a software solution to improve overall performance (generally 5% of software maintenance costs).
- Enhancements—The costs due to continuing innovations (generally 50% or more of software maintenance).

Maintenance Effort by Year

Base Year: 2009

Fiscal Year Start Month: 1

Fiscal Year	Average Staff Level	Effort Months Correct	Adapt	Perfect	Enhance	Total	Cumulative	Base Year Cost
2008	4.8	15.1	4.1	23.5	4.7	47.4	47.4	592,482
2009	4.3	12.5	7.1	20.3	11.7	51.7	99.1	645,732
2010	3.4	3.9	9.3	10.5	17.1	40.9	139.9	510,672
2011	3.2	2.7	9.5	9.1	17.6	38.8	178.7	485,257
2012	3.2	2.7	9.5	9.1	17.6	38.8	217.6	485,257
2013	3.2	0.5	1.7	1.6	3.1	6.8	224.4	85,028

Figure 4. SEER-SEM Maintenance Effort by Year Report
(Reifer, Allen, Fersch, Hitchings, Judy, & Rosa, 2010)

SEER–SEM requires the developer to contribute inputs to the model based on a set of parameters associated with the anticipated sustainment attributes of the software. For example, the category Maintenance Growth Over Life contains a rating correlated to how much software growth the customers anticipate once the maintainers receive the software in the maintenance cycle, as indicated in Table 3. A developer can assume that once the software goes into the maintenance cycle, "an input of 100% means that the software will double in size" (Galorath Incorporated, 2001, pp. 7–55).

Table 3. SEER-SEM Maintenance Growth Over Life Parameters
(Galorath Incorporated, 2001)

Rating	Description
100%	Very high, major updates adding many new functions
35%	High, major updates adding some new functions
20%	Nominal, minor updates with enhancements to existing functions
5%	Low, minor enhancements
0%	Very low, sustaining engineering only

Other parameters that can be included to derive a software maintenance report are years of maintenance, annual change rate, differences in the development environment, maintenance level (rigor), and maintenance monthly labor rate (Galorath Incorporated, 2001).

SEER–IT differs from SEER–SEM in that SEER–IT extends beyond the software and examines a proposed (or purchased) “IT system’s services, infrastructure and risk for the project and ongoing support” (Galorath Incorporated, 2011a). The scope of SEER–IT is much broader than SEER–SEM in order to include the ability to build project portfolios that allow managers to estimate return on investment (ROI) for particular IT projects. By drawing on historical databases of several previous IT projects provided by Galorath, SEER–IT is able estimate the maintenance costs for an IT project (considered on-going support) based on the data provided by the customer, as shown in Figure 5. The combination of these estimation tools would provide a great deal of insight into the projected cost of software maintenance and associated IT projects.

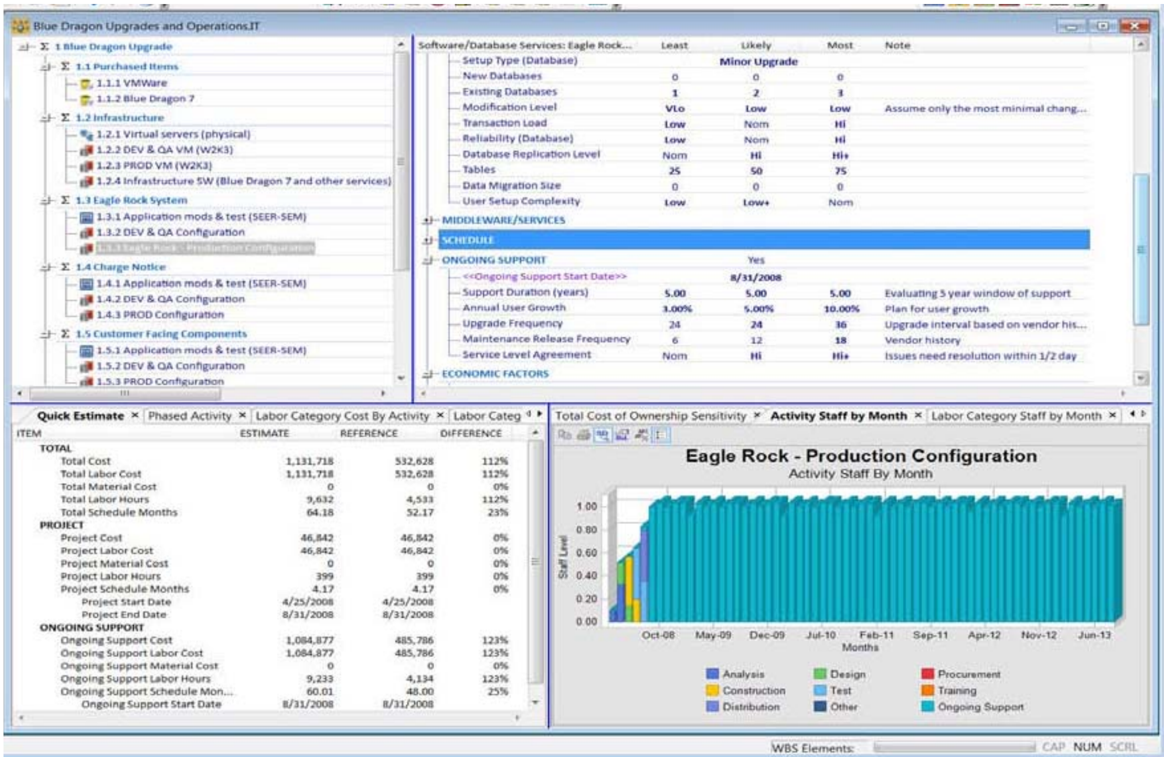


Figure 5. SEER–IT On-Going Support Example
(Reifer et al., 2010)

4. Software Lifecycle Management (SLIM)–Suite of Tools

Developed by Quantitative Software Management (QSM) Incorporated, Software Lifecycle Management (SLIM) contains several products that create reports, graphs, and forecasts in order to defend software projects. SLIM–Estimate is just one product from the SLIM suite designed to provide solutions to complex problems facing project managers or developers. Other products include the following: SLIM–Control, SLIM–Metrics, SLIM–DataManager and SLIM–MasterPlan (QSM, 2006). SLIM–Estimate allows the customer to import his or her own data from previous projects in order to calibrate the SLIM estimate (similar to SEER–SEM and SEER–IT), or the customer can choose to employ the SLIM historical database to provide more data points in the estimation.

SLIM–Estimate breaks development into four distinct phases typically associated with the software development life cycle. These phases are as follows: (1) Concept Definition, (2) Requirements and Design, (3) Construction and Test, and (4) Perfective

Maintenance (QSM, 2006). QSM (2006) defined maintenance as “correcting errors revealed during system operation or enhancing the system to adapt to new user requirements, changes in the environment, and new hardware” (QSM, 2006, p. 78). SLIM–Estimate addresses software maintenance in the project environment portion of the model in the perfective maintenance tab. The maintenance inputs of the SLIM–Estimate model can then be transferred to the additional SLIM–MasterPlan tool to produce an easy-to-read display, as shown in Figure 6. In this case, Figure 6 demonstrates the estimated expected costs of a simulated software maintenance project over a three-year period. This report includes major and minor enhancements as well as other maintenance associated tasks within the Baseline Support category (i.e., emergency fixes and help desk support). This model provides program managers defendable position from which to justify manpower increases/decreases as displayed in man-months (MM), and budget requests, as exhibited in the ($1,000) column.

Program Summary Report
Maintenance Model 2

Task	Task Description	Start Date	End Date	Elapsed Months	MM	($1000)
TOTAL SUPPORT	Summary Task	1/1/2008	2/23/2011	37.82	1,445.23	23,413
MAJOR ENHANCEMENTS	Summary Task	1/1/2008	2/23/2011	37.82	856.14	13,870
Major Enhancement-Y1	SLIM-Estimate Subsystem (Majo...	1/1/2008	4/23/2009	15.77	285.38	4,623
Major Enhancement-Y2	SLIM-Estimate Subsystem (Majo...	11/30/2008	3/21/2010	15.71	285.38	4,623
Major Enhancement-Y3	SLIM-Estimate Subsystem (Majo...	10/31/2009	2/23/2011	15.85	285.38	4,623
MINOR ENHANCEMENTS	Summary Task	1/1/2008	12/31/2010	36.00	157.09	2,545
Minor Enhancement 1-Y1	SLIM-Estimate Subsystem (Mino...	1/1/2008	5/13/2008	4.42	13.61	221
Minor Enhancement 2-Y1	SLIM-Estimate Subsystem (Mino...	3/31/2008	8/9/2008	4.32	13.05	211
Minor Enhancement 3-Y1	SLIM-Estimate Subsystem (Mino...	6/27/2008	11/4/2008	4.27	13.09	212
Minor Enhancement 4-Y1	SLIM-Estimate Subsystem (Mino...	9/23/2008	1/31/2009	4.27	13.11	212
Minor Enhancement 1-Y2	SLIM-Estimate Subsystem (Mino...	12/19/2008	4/25/2009	4.25	13.03	211
Minor Enhancement 2-Y2	SLIM-Estimate Subsystem (Mino...	3/13/2009	7/21/2009	4.29	13.04	211
Minor Enhancement 3-Y2	SLIM-Estimate Subsystem (Mino...	6/9/2009	10/16/2009	4.25	13.04	211
Minor Enhancement 4-Y2	SLIM-Estimate Subsystem (Mino...	9/4/2009	1/11/2010	4.25	13.04	211
Minor Enhancement 1-Y3	SLIM-Estimate Subsystem (Mino...	11/30/2009	4/7/2010	4.27	13.01	211
Minor Enhancement 2-Y3	SLIM-Estimate Subsystem (Mino...	2/24/2010	7/2/2010	4.24	13.04	211
Minor Enhancement 3-Y3	SLIM-Estimate Subsystem (Mino...	5/22/2010	10/1/2010	4.35	13.03	211
Minor Enhancement 4-Y3	SLIM-Estimate Subsystem (Mino...	8/21/2010	12/31/2010	4.35	13.02	211
BASELINE SUPPORT	Summary Task	1/1/2008	2/23/2011	37.82	432.00	6,998
Emergency Fixes	Custom Task	1/1/2008	2/23/2011	37.82	31.42	509
Infrastructure upgrades	Custom Task	1/1/2008	2/23/2011	37.82	78.54	1,272
Help Desk	Custom Task	1/1/2008	2/23/2011	37.82	196.36	3,181
Operational Support	Custom Task	1/3/2008	2/25/2011	37.83	102.11	1,654
Research Projects	Custom Task	1/3/2008	2/25/2011	37.83	23.56	382
Overall Program		1/1/2008	2/23/2011	37.82	1,445.23	23,413

Figure 6. SLIM Maintenance Screen
(Reifer et al., 2010)

5. Summary

COCOMO II, SEER–SEM, SEER–IT, and SLIM–Estimate all provide program managers with an appropriate amount of information necessary to estimate the costs of software maintenance for a given program or project. These models "assume that software maintenance is a subset of development, not the opposite" (Reifer et al., 2010, p. 10). Using these models, developers and program managers are able to adjust the cost factors and continue to refine their calibration of whichever model they employ.

THIS PAGE INTENTIONALLY LEFT BLANK

III. DATA AND METHODOLOGY

> Software not developed with maintenance in mind can end up so poorly designed and documented that total redevelopment is actually cheaper than maintaining the original code.
>
> – Department of the Air Force (2000)

A. SAMPLE DATA SET USED DURING RESEARCH

Data for this thesis was collected from the Office of the Secretary of Defense Cost and Resource Center (DCARC) and compiled by the NAVAIR to support local ongoing research. The majority of the data obtained for this thesis was graciously provided by Dr. Wilson Rosa of the Information Technology Division of the AFCAA and Mr. Peter Braxton of Technomics Incorporated. The AFCAA and Technomics are currently conducting an Air Force–sponsored study on software maintenance and were able to provide the results of their collection efforts thus far to support this thesis. Their study's objectives are to collect "actual data to improve software maintenance cost estimating" (Rosa & Braxton, 2010). The results of the AFCAA study are to support better cost-estimating techniques and to provide benchmarks for both industry and government agencies that can be used in future proposals (Rosa & Braxton, 2010). A data item description (DID) was provided to various contractors and government agencies for them to complete and return to Technomics for inclusion in the study's database. The final DID that was provided to the data sources can be found in Appendix A. However, agencies and industry partners submitted data prior to the completion of the DID; therefore, this data was not normalized to match categories required by the DID. The normalization process is currently being conducted by Technomics. Nevertheless, the AFCAA and Technomics were able to provide whatever raw data they had available.

1. Warner Robins Air Logistics Center

In the summer of 2009, Reifer Consultants, Inc., conducted a software maintenance study that involved various government agencies. Warner Robins Air Logistics Center (ALC) was one such agency. ALC personnel who were working on a variety of projects at equally varying times in the acquisition cycle completed questionnaires in support of the study. Based on the information provided in the questionnaires, the participants were selected for further interviews. If any additional interviews were conducted, this data was not available.

From the set of eight available questionnaires, seven programs were selected due to the completeness of the information provided and the applicability to this thesis. Each questionnaire was completed by program managers, leads, software managers, or integrated product team (IPT) leads. The range of programs from those selected reported avionics as their operating environment. The questionnaires indicted the various programming languages used in their software, as shown in Table 4.

Table 4. Warner Robins ALC Programs and Languages

Program	Primary Software Language
Joint Stars	C/C++
MC-130E Combat Talon	Jovial J73
MMRT BCC-001	Ada
MRT E20	Ada
SOF EISE Sustainment	Ada
USAF F-15 Suite S7E Block Upgrade	Jovial
ALR-56M Block Cycle D	Access

The application domains stated in the questionnaires included electronic warfare, command and control, radar and weapons delivery, and database (which included simulation and modeling as well as controls and displays). Other information contained in the questionnaires included software change request information, the activities included in the effort (divided between software maintenance and sustaining engineering), and the success rating for the project. Next, the questionnaire inquired about the actual resources expended/estimated (for completed software projects). This allowed the program managers to record their cost estimates and drivers during the development of releases. These were documented through the categories of Total Resources Expended, Resource Allocations (Labor Hour by Major Activity), Size Information, and Modified Code. Lastly, the questionnaire enabled participants to indicate scale factor ratings as designed by the COCOMO II model. The program managers were able to indicate the estimated rating and the actual value of the scale factor at completion.

The researcher transferred this raw data to an Excel spreadsheet for convenience and ease of analysis. The data was categorized by source lines of code (SLOC), costs, and the percentage of maintenance effort applied in the software release (whether adaptive, corrective, perfective, or enhancements). Additionally, three programs were able to report their budgeted and actual cost of release by the number of hours applied to the project.

2. Picatinny Arsenal

Data from the Picatinny Arsenal was obtained by the AFCAA through the Office of the Deputy Assistant Secretary of the Army for Cost and Economics (ODASA–CE) and normalized by Technomics into the DID spreadsheet mentioned earlier. The data set contained a total of 19 projects from four programs (the Light Weight Mortar Ballistic Computer, the Mortar Fire Control System—Heavy, the Paladin system, and the Towed Artillery Digitization) at various versions or software blocks. The researcher selected seven programs from the available data due to the completeness of information provided and the applicability to this thesis. The final candidate projects used for this thesis and their associated programming languages are listed in Table 5.

Table 5. **Picatinny Arsenal Programs and Languages**

Program	Primary Software Language
LHMBC Version 3	C++
Paladin SWB2 Version 3	Ada
Paladin SWB2 Version 2	Ada
Paladin V7P	Ada
Paladin V7	Ada
Paladin V11.4	Ada
TAD Block 1A	C++

The researcher transferred this raw data to an Excel spreadsheet for convenience and ease of analysis. The data was then categorized by a summarized tabulation of SLOC (divided by deleted, modified, new, and reused) and overall costs. Additionally, one program reported the number of defects categorized by priority of the defect. This data point was also included in the Excel spreadsheet.

3. Integrated Strategic Planning and Analysis Network

As described in the DoD's 2008 *Major Automated Information System Annual Report*, the Integrated Strategic Planning and Analysis Network (ISPAN) Block I employs a

> system of systems approach that spans multiple security enclaves for strategic and operational level planning and leadership decision making. The system is composed of two elements: (1) a Collaborative Information Environment (CIE) managing strategy-to-execution planning across all United States Strategic Command (USSTRATCOM) Mission areas; and (2) a Mission Planning and Analysis System (MPAS) that support the development of Joint Staff Level I through Level IV nuclear and conventional plans supporting National and Theater requirements. (DoD, 2008)

The data provided to the AFCAA included several years' worth of development and maintenance information related to the suite of ISPAN programs. "The major application software programs used in the process (ISPAN) include the National Ground Zero Integrated List and Development System (NIDS), the Missile Graphics Planning System (MGPS), the Air Vehicle Planning System (APS), and the Document Production System (DPS)" (United States Strategic Command [USSTRATCOM], 2004, p. 2).[1] Additional programs included the Automated Windows Planning System (AWPS), the Theater Integrated Planning System (TIPS), and others related to the ISPAN program. This information was divided by SLOC and Software Change Requests and then further segregated by the major programs within ISPAN. The various projects and their associated programming languages used for this thesis are depicted in Table 6.

Table 6. ISPAN Programs and Languages

Program	Primary Software Language
Automated Windows Planning System (AWPS)	C
Missile Graphics Planning System (MGPS)	FORTRAN
Aircraft Air Vehicle Planning System	C++
Data Services	C/C++
Theater Integrated Planning System (TIPS)	Unknown
National Ground Zero Integrated List and Development System (NIDS)	C++

The ISPAN data revealed the acquisition method used for each subordinate program. This information was broken down into two categories, custom build or COTS purchase. Additionally, the labor effort performed (by percentage) was partitioned between three categories: adaptive, perfective, and corrective. Finally, the ISPAN data contained full-time equivalent (FTE)_for maintenance personnel (between 2003 and

[1] This document was provided to the researcher by the AFCAA for inclusion in a study on software maintenance.

2008), segregated by each major subordinate program, as well as the logical source lines of code for these programs.

4. Lockheed Martin Systems Integration Owego

The Lockheed Martin data provided to the AFCAA arrived without an appropriate data dictionary for use in sorting out the various category definitions listed in the Excel spreadsheet provided. However, simple deduction and common assumptions permitted the use of the data. The information Lockheed Martin gave on several of its programs provided three years' worth of aviation-related software maintenance. These programs performed a variety of services, including built-in-testing and common console applications. The software types themselves were split between support and embedded software. The programs and their associated programming languages are displayed in Table 7.

Table 7. Lockheed Martin Systems Integration Owego Programs and Languages

Program	Primary Software Language
CDNMDLT_IMOP_MHP	Java
ESM MHP BIT	C++
ESM MMH BIT	C++
JAGRS–Total	C
CP140 IMOP Emulator R4.0–Total	Ada
MMH ESM OFP MERGE SW–Total	Ada
MMH LASIS 15.5, 15.6, 15.7 & 15.8–Total	Ada
MMH P3I Dev Rel 15	Ada
SBC Legacy BSP R11–Total	C
VH-71 VASIS 5.0–Total	Ada
MMH-P3I AOP SW	Ada
MMH LASIS 15.9 & 17.0–Total	Ada
AMCM Common Console–Total	C
A10_PE_ISA	C#

The data contained whether the software underwent maintenance while being developed or whether it reflected only maintenance actions on those programs. This data also held the start and end dates for any maintenance that was performed. The range for these dates varied from as short as three months to as long as six years. SLOC counts were recorded by base code, automatically generated code, modified, new, reuse, ported, and their aggregate totals. Additionally, the data contained the number of defects reported across several categories.

5. Naval Air Systems Command (NAVAIR)

A portion of the data provided by the NAVAIR 4.2 Cost Department was the result of a previous analysis conducted on several software-intensive programs and their associated information contained within the software resources data report (SRDR). NAVAIR collected this data over several months via the Defense Cost and Resource Center (DCARC) website to discover any trends related to the development language and the type of software being created, reused, modified, or automatically generated. The primary documents used to derive the Excel spreadsheet provided were taken by the NAVAIR Cost Department from the SRDR (either the 2630-2 or 2630-2) for that particular program. There were well over 1,300 data points from 47 disparate programs identified in the data. However, NAVAIR reported that many data points were considered unreliable for analysis: “In working with the data we recognized that some of the actual data points were not very meaningful, either they were an interim build actual that was not stand alone or the data turned in was highly questionable.”[2] The extensive amount of information contained in NAVAIR’s analysis precipitated the need to limit the data used for this thesis to 16 data points associated with nine programs, as shown in Table 8.

[2] This information can be found in the database received from NAVAIR 4.2 Cost Department under the tab titled *Filter Tips* in the Microsoft Excel spreadsheet titled 2630 Raw Sep 10.xls.

Table 8. Naval Air Systems Command SRDR Study Programs and Languages

Program	Primary Software Language
AEA Mission Planning SW Build 1&2	Visual basic
Operational Flight Program SW Build 1&2 Final	Ada
AN/USG-2/3 CEC DDS Tactical CSCI	Ada
Intelligent Services Build 1 End	C++
I/O Services Build 1 End	C++
System of Systems Common Operating Environment (SOSCOE) Build 1.5 Final	C++
SCS 4.0 Mission Computer	Ada
F-16 Block 30 SCU 7 UPC	C#
Apache Longbow Block III	Ada
Active Controls (First Flight)	C (ANSI C)
AHE Mission Computer Build 2 (Release 0)	Ada
AHE Mission Computer Support Build 2 (Release 0)	C/C++
AHE Mission Display Build 2	C++
AHE Comm Suite (UTFA1/UTFA3)	C/Assembly
AHE Radar (AN/APY-9)	C/C++
Mission Support SW Initial Release	Java

NAVAIR’s collection of data provided information from the SRDRs of these programs through SLOC counts and categorized by base code, new, modified, reused, and automatically generated code. Additionally, the data collection identified the software developer and its self-reported Capability Maturity Model—Integrated (CMMI) maturity levels. Lastly, the data provided the time taken to develop the software and the contractor’s overall productivity in relation to the SLOC type reported (new, modified, unmodified).

The remaining portion of the data obtained from NAVAIR also included information from 61 software projects and their related Program Related Engineering

(PRE) costs. PRE is a program of record that provides software support to the tactical software systems for the Navy and Marine Corps (NAVAIR, 2010). The funding for PRE is divided between Capability Defect Package (CDP) and Fleet Response Activity (FRA). The CDP collects software trouble reports, performs analysis of these reports, and then delivers the software to the operating forces (NAVAIR, 2010). FRA funds are used by the Software Support Activity (SSA) for any other resources that are not identified as CDP. This data set included several years' worth of PRE actual amount funded (from 1995 to 2008) and expected funding (from 2009 to 2015) for these programs. Additionally, the data set included major program subsystems/CSCIs, the number of units/subsystems deployed to users, information concerning the maintainer (name, CMM and CMMI levels), and the SLOC for the associated subsystems/CSCIs. These 61 candidate programs lacked consistency for the program's actual amount funded; therefore, the researcher narrowed the programs to those that possessed five consecutive years' worth of PRE actual amount funded data. NAVAIR arranged this data by the SLOC for each programs' subsystems/CSCIs. The researcher combined the total SLOC and number of subsystems/CSCIs for ease of analysis. Additionally, the researcher averaged the number of units/systems deployed to users. Unfortunately, the programming language was not contained in the PRE data set. The total used for this research was 28 programs. The programs represented in the data were divided into five groupings determined by their functions or by the major hardware they supported. These categories are air combat equipment (ACE), aviation support equipment (ASE), missiles (MIS), fixed wing aviation (FWA), and rotary wing aviation (RWA). The software product teams' programs and their associated application domain used in this research are shown in Table 9.

Table 9. Naval Air Systems Command PRE Software Product Team Programs and Application Domains

Software Product Team	Domain	Software Product Team	Domain
PMA170_GPS/CDNU	ACE	PMA265_F/A18	FWA
PMA209_AMC&D	ACE	PMA271_E6B	FWA
PMA209_CAINS	ACE	PMA273_T45	FWA
PMA209_GPSW-TAWS	ACE	PMA290_P3C	FWA
PMA209_CSFIR	ACE	PMA242_HARM	MIS
PMA209_SDRS	ACE	PMA259_AIM9X	MIS
PMA209_TAMMAC	ACE	PMA259_AMRAAM	MIS
PMA260_CASS	ASE	PMA226_H46	RWA
PMA272_EWSSA	ASE	PMA261_H53	RWA
PMA207_C-130 F,R&T	FWA	PMA275_V22	RWA
PMA231_E2-C	FWA	PMA276_AH1W	RWA
PMA207_KC130J	FWA	PMA276_UHIN	RWA
PMA234_EA6B/AEA	FWA	PMA299_H60B-LAMPS	RWA
PMA257_AV8B	FWA	PMA299_H60FH	RWA

B. VARIABLES AND METHODOLOGY

The disparate number of variables, lack of consistency, and normalization across the data limited the ability to perform extensive multivariate regression analysis across the data collected. The researcher could not assure that any result from performing traditional multivariate analysis would reveal the desired cost-estimating relationship needed to create a cost model for software maintenance as originally intended. Instead, the statistical tool JMP (Release 9) produced by the SAS Institute was used to derive the analysis for this thesis. This package was principally chosen due to its availability to NPS students for free. Additionally, JMP contains data tables that are easily converted and manipulated from Excel spreadsheets. Additionally, JMP produces visually attractive graphical material for analysis. This was compared to Excel, where the researcher needed to create several different tabs in order to analyze a single data set, and the graphical choices were limited. The variables selected for correlations or regressions were chosen depending on the integrity of the data available and on assumptions concerning cost drivers for software maintenance. Some of the variables chosen were SLOC types, overall cost, effort types (adaptive, corrective, perfective), number of software change requests, total number of defects reported, and the number of FTEs for a particular year's worth of maintenance. Any cost-related values were retained within their reported fiscal years for consistency and not converted to reflect inflation.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. DATA ANALYSIS

> Calculating maintenance costs is a multi-dimensional problem and the software itself is only one of the many dimensions of that problem. There is not only a product to be maintained, but also a maintenance process, a maintenance environment, maintenance personnel and the tools available.
>
> – Harry M. Sneed (2004)

A. CORRELATION ANALYSIS

1. Purpose

The data analysis for this thesis began with simple correlations between the variables collected within the data provided. This test was important because it allowed the researcher to understand the linear relationship between two variables. The formula for the simple Pearson product-moment correlation is represented in Equation 5.

$$r_{XY} = \frac{n\sum XY - \sum X \sum Y}{\sqrt{\left[n\sum X^2 - \left(\sum X\right)^2\right]\left[n\sum Y^2 - \left(\sum Y\right)^2\right]}} \tag{5}$$

where r_{XY} is the correlation coefficient between X and Y, and

n is the size of the sample,

X is the X variable,

Y is the Y variable,

XY is the product of the X variable multiplied by the corresponding Y variable,

X^2 is the X variable squared, and

Y^2 is the Y variable squared. (Salkind, 2004, p. 81)

For the purposes of this thesis, the correlation coefficient was used to determine which pairing between variables contained the strongest relationships. The results of this analysis were then used to extract candidate variables to compute simple linear regressions and possibly create cost-estimating relationships.

2. Warner Robins and ISPAN Data Analysis

The data set provided by Warner Robins did not analyze well for this thesis. The information provided did not contain enough cost data for analysis to calculate correlations alone. However, the data set did provide a basis for comparing the amount of SLOC compared to the maintenance effort applied. This information was also contained in the ISPAN program data. Therefore, these two data sets (totaling nine programs) were combined in order to analyze their results categorically by software size and effort type (corrective or perfective maintenance). It was assumed that the amount of maintenance performed would correlate to the complexity of the software, but since there were no metrics provided that could be used as a surrogate for complexity, software size was computed as the dependent variable for analysis.

The correlation between the variables' total source lines of code and percentage effort of corrective and perfective maintenance resulted in the report shown in Figure 7. The outcome demonstrated that for this combined data set, the percentage of effort in perfective maintenance correlated (0.75) to the source lines of code (depicted as Total SLOC). However, the percentage of effort in corrective maintenance showed a negative correlation (-0.63) to the source lines of code. Therefore, complexity could not be definitively proven by the percentage effort of maintenance performed and the total amount of SLOC in the software.

Correlations

	Total SLOC	Effort (Corrective)	Effort (Perfective)
Total SLOC	1.0000	-0.6331	0.7558
Effort (Corrective)	-0.6331	1.0000	-0.4383
Effort (Perfective)	0.7558	-0.4383	1.0000

Figure 7. Multivariate Correlation Results for SLOC and Percentage of Maintenance Effort for SW Programs

3. Picatinny Arsenal Data Analysis

This data appeared to be the most promising toward building a software maintenance model. This assumption was based on actual cost information and descriptions of the software reported in the collected data.

The first correlation resulted in the report shown in Figure 8. The outcome demonstrated that for this data set, the original base count of source lines of code (depicted in Figure 8 as SLOC Reused [Old]) has little correlation (0.28) to the overall costs associated with the maintenance. However, the number of SLOC introduced to the base code (depicted in Figure 8 as SLOC [Added]) resulted in a strong correlation (0.81) to the overall cost of the maintenance.

Correlations

	Overall Costs	SLOC New (Added)	SLOC Reused (Old)	Total Delivered SLOC
Overall Costs	1.0000	0.8138	0.2824	0.6310
SLOC New (Added)	0.8138	1.0000	-0.2560	0.1296
SLOC Reused (Old)	0.2824	-0.2560	1.0000	0.9214
Total Delivered SLOC	0.6310	0.1296	0.9214	1.0000

Figure 8. Multivariate Correlation Results for Cost and SLOC

The Picatinny Arsenal data also included the total effort (in man-months) used for the maintenance. This data could be used as a proxy for dollar costs. The results of the correlation for this variable with SLOC counts are shown in Figure 9. The total effort variable was not strongly correlated (0.24) to the amount of SLOC reused in the maintenance. However, SLOC (Added) continued to show a strong correlation (0.73) compared to the total effort variable.

Correlations

	Total Effort (man hours)	SLOC New (Added)	SLOC Reused (Old)	Total Delivered SLOC
Total Effort (man hours)	1.0000	0.7360	0.2468	0.5787
SLOC New (Added)	0.7360	1.0000	-0.3066	0.0858
SLOC Reused (Old)	0.2468	-0.3066	1.0000	0.9178
Total Delivered SLOC	0.5787	0.0858	0.9178	1.0000

Figure 9. Multivariate Correlation Results for Total Effort and SLOC

This data also included the requirements added or deleted for the particular software represented. However, the Paladin SWB2 (version 3) was excluded for analysis because it did not include information for either one of these variables. The results of the five data points and their associated variables are shown in Figure 10. These analyses revealed that there were no strong correlations between the requirements added (new to the version or release, represented in Figure 10 as Reqts(+)) or deleted (existing requirements deleted from a previous release or version, represented in Figure 10 as Reqts(-)) and the overall cost of the maintenance performed.

Correlations

	Overall Costs	Reqts (+)	Reqts (-)	Deliv'd SLOC
Overall Costs	1.0000	-0.0357	-0.0162	0.6310
Reqts (+)	-0.0357	1.0000	-0.1008	-0.5626
Reqts (-)	-0.0162	-0.1008	1.0000	-0.4956
Deliv'd SLOC	0.6310	-0.5626	-0.4956	1.0000

Figure 10. Multivariate Correlations Report for Cost and Requirements

The same analysis was conducted for total effort against these variables, as shown in Figure 11. This analysis also revealed that there were no strong correlations between the requirements added or deleted, and the total effort contributed to the software maintenance.

Correlations

	Total Effort (man hours)	Requirements Added	Requirements Deleted	Total Delivered SLOC
Total Effort (man hours)	1.0000	-0.0803	0.0625	0.5705
Requirements Added	-0.0803	1.0000	-0.1008	-0.5626
Requirements Deleted	0.0625	-0.1008	1.0000	-0.4956
Total Delivered SLOC	0.5705	-0.5626	-0.4956	1.0000

Figure 11. Multivariate Correlations Report for Total Effort and Requirements

4. Integrated Strategic Planning and Analysis Network Data Analysis

This data set provided six years' worth of logical SLOC, the FTE associated with the maintenance conducted on ISPAN's subprograms, the number of CSCIs associated with those subprograms, and the maintenance defect count for four years (2005–2008). Since actual cost data was not provided in the data set, it was assumed that FTE data could be used as a surrogate. The number of CSCIs listed in the data set indicated that they did not change from year to year; therefore, the number of CSCIs was held constant in the analysis. These numbers were then correlated by year, as shown in Figures 12 and 14. The remaining reports for fiscal years 2006 and 2007 are located in Appendix B.

Correlations

	FTE Maintenance	SLOC	Defects	CSCIs
FTE Maintenance	1.0000	0.8602	0.3645	0.1465
SLOC	0.8602	1.0000	0.4321	-0.0295
Defects	0.3645	0.4321	1.0000	0.0513
CSCIs	0.1465	-0.0295	0.0513	1.0000

Figure 12. Multivariate Correlations Report for FY05 ISPAN Data

The results show that SLOC and the number of FTEs for maintenance contain the strongest correlation for FY05. Since one subprogram contained a singular CSCI, the researcher determined that this could skew the results of the correlation and recalculated the correlation; the results are shown in Figure 13. However, these results did not significantly improve the relationship between the proxy for cost (FTE Maintenance) and the number of CSCIs in the FY05 ISPAN program.

Correlations

	FTE Maintenance	SLOC	Defects	CSCIs
FTE Maintenance	1.0000	0.9667	0.3985	-0.3110
SLOC	0.9667	1.0000	0.4006	-0.2442
Defects	0.3985	0.4006	1.0000	-0.0152
CSCIs	-0.3110	-0.2442	-0.0152	1.0000

Figure 13. Multivariate Correlations Report for FY05 ISPAN Data Minus One Subprogram With a Singular CSCI

The analysis of the ISPAN data set from FY08 revealed similar results as FY05, as shown in Figure 14. The number of CSCIs continued to be less of a factor, contributing to the amount of FTE maintenance performed on the software.

Correlations

	FTE Maintenance	SLOC	Defects	CSCIs
FTE Maintenance	1.0000	0.7148	0.4650	0.0195
SLOC	0.7148	1.0000	0.5535	0.0289
Defects	0.4650	0.5535	1.0000	-0.2937
CSCIs	0.0195	0.0289	-0.2937	1.0000

Figure 14. Multivariate Correlations Report for FY08 ISPAN Data

In order to ensure that the singular CSCI count for one subprogram did not influence the results, another correlation was performed minus that particular program. The results are shown in Figure 15. As expected, the CSCI count did not reflect any relationship to the amount of FTE maintenance. However, the correlation between the amount of FTE maintenance and defects rose considerably from 0.46 to 0.86.

Correlations

	FTE Maintenance	SLOC	Defects	CSCIs
FTE Maintenance	1.0000	0.5728	0.8663	-0.6204
SLOC	0.5728	1.0000	0.6431	-0.2185
Defects	0.8663	0.6431	1.0000	-0.3265
CSCIs	-0.6204	-0.2185	-0.3265	1.0000

Figure 15. Multivariate Correlations Report for FY08 ISPAN Data Minus One Subprogram With a Singular CSCI

5. Lockheed Martin Systems Integration Data Analysis

This data set mostly contained information from FY07, but also it included data from FY08 and one program's data for FY09. The Lockheed Martin data included the start and end date of the maintenance performed on these programs. The number of months contained in this information was calculated and analyzed to determine if this data was related to the number of labor months. The result was a 78% correlation. Since the data did not include actual cost data, the number of labor months was used as a proxy to determine cost factors in the remainder of the correlation analysis.

The analysis of this data revealed that the strongest correlation was between the number of labor months and the modified code (0.83), as shown in Figure 16. Not surprisingly, a strong relationship exists between modified code and the number of defects. This implies that the amount of modified code increases with the number of defects in the software. However, the second strongest relationship is between defects and labor months.

Correlations				
	Labor Months	Total Defects	Base Code	Modified Code
Labor Months	1.0000	0.6553	0.3275	0.8256
Total Defects	0.6553	1.0000	0.5902	0.6443
Base Code	0.3275	0.5902	1.0000	-0.0476
Modified Code	0.8256	0.6443	-0.0476	1.0000

Figure 16. Multivariate Correlations Report for Multiyear Lockheed Martin Data for Labor Months, Defects, Modified, and Base Code

Next, an analysis of the amount of new and reused code was performed, as shown in Figure 17. As expected, the amount of new code introduced had a very high correlation (0.95) to the amount of labor months used in the maintenance. The amount of reused code was significantly lower (-0.17) than anticipated because there were only two programs that reported reuse code numbers, which influenced the lower correlation.

Correlations			
	Labor Months	New Code	Reused Code
Labor Months	1.0000	0.9585	-0.1704
New Code	0.9585	1.0000	-0.1864
Reused Code	-0.1704	-0.1864	1.0000

Figure 17. Multivariate Correlations Report for Multiyear Lockheed Martin Data for Labor Months, New, and Reused Code

6. NAVAIR Program Related Engineering (PRE) Data Analysis

This data was analyzed to extract the most complete information possible concerning size of the software (SLOC), the number of associated subsystems or CSCIs, the number of deployed systems that use the software, and the amount funded for that program for a particular year. The data was then narrowed down to those programs that contained funded PRE data for at least five consecutive years. Once this funding criterion was met, the total number of program CSCIs was computed as well as the associated SLOC. Finally, the number of deployed units or subsystems within a program was averaged. This was done to account for the support activity's inability to conduct maintenance on every single piece of equipment within that particular year's worth of PRE funds. It was assumed that some of the software maintenance would carry over to the next year's funding. Therefore, the researcher determined that it was more appropriate to average the amount of units/subsystems deployed for the purposes of this research.

The actual PRE funded amounts varied by year as well as by category. The programs represented in the data were divided into five groupings, determined by their functions or by the major hardware they supported. These categories were air combat equipment (ACE), aviation support equipment (ASE), missile systems (MIS), fixed wing aviation (FWA), and rotary wing aviation (RWA), as shown in Table 10. It appears that the vast majority of PRE funding is spent in support of fixed wing aviation, as shown in Figures 18 and 19, which display the FY04 and FY08 summation amounts funded by category. The charts for the remaining fiscal years can be found in Appendix B. However, when the mean of these amounts was computed for the identical years, aviation support equipment dominated PRE funding, as shown in Figures 20 and 21. The amount of funding is mentioned only to establish the background for the remainder of the data analysis on the information provided by NAVAIR.

Table 10. NAVAIR PRE Data Categories

Category	Abbreviation
Air Combat Equipment	ACE
Aviation Support Equipment	ASE
Fixed Wing Aviation	FWA
Missile Systems	MIS
Rotary Wing Aviation	RWA

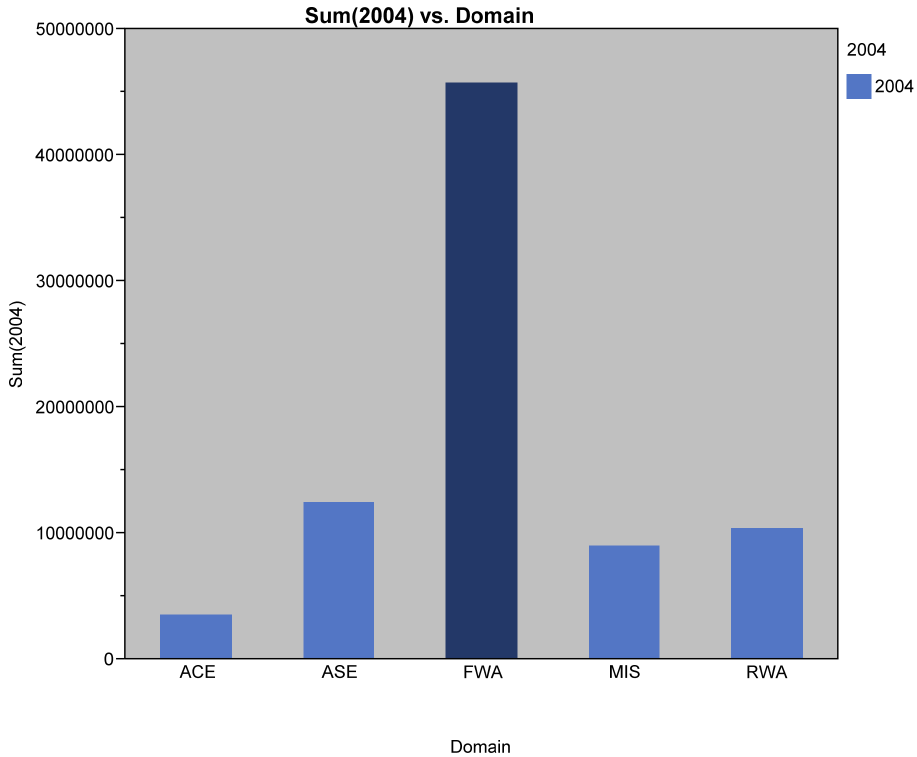


Figure 18. Sum of PRE Actual Funded Amount for FY04 by Category

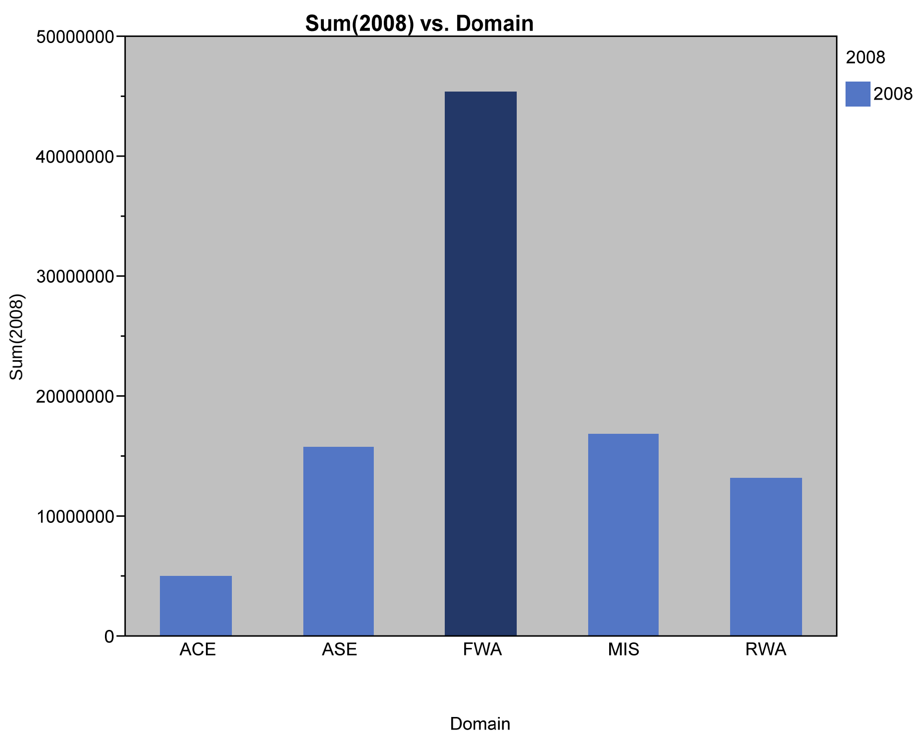


Figure 19. Sum of PRE Actual Funded Amount for FY08 by Category

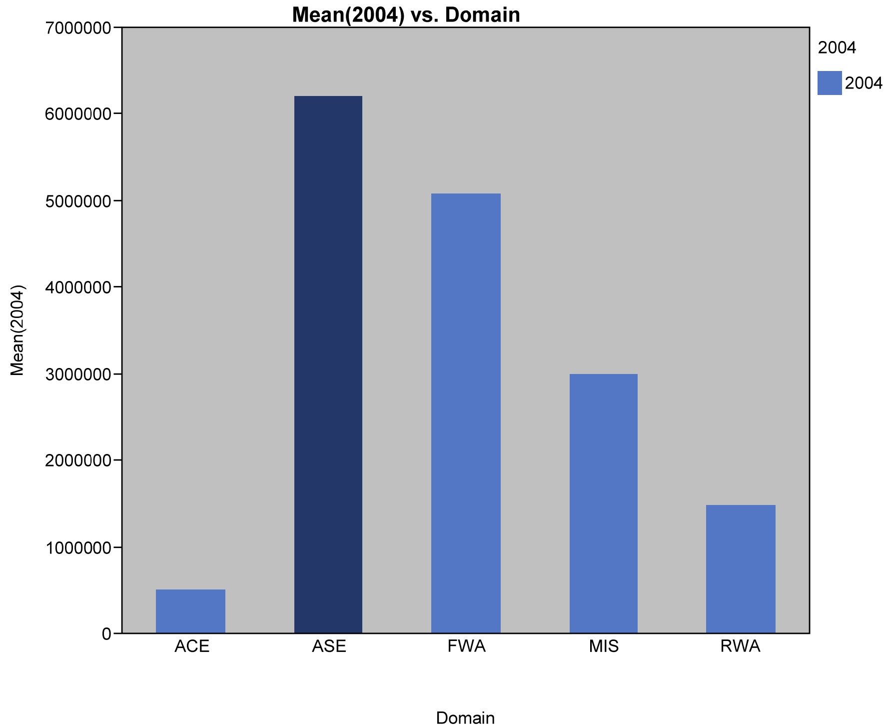


Figure 20. Mean of PRE Actual Funded Amount for FY04 by Category

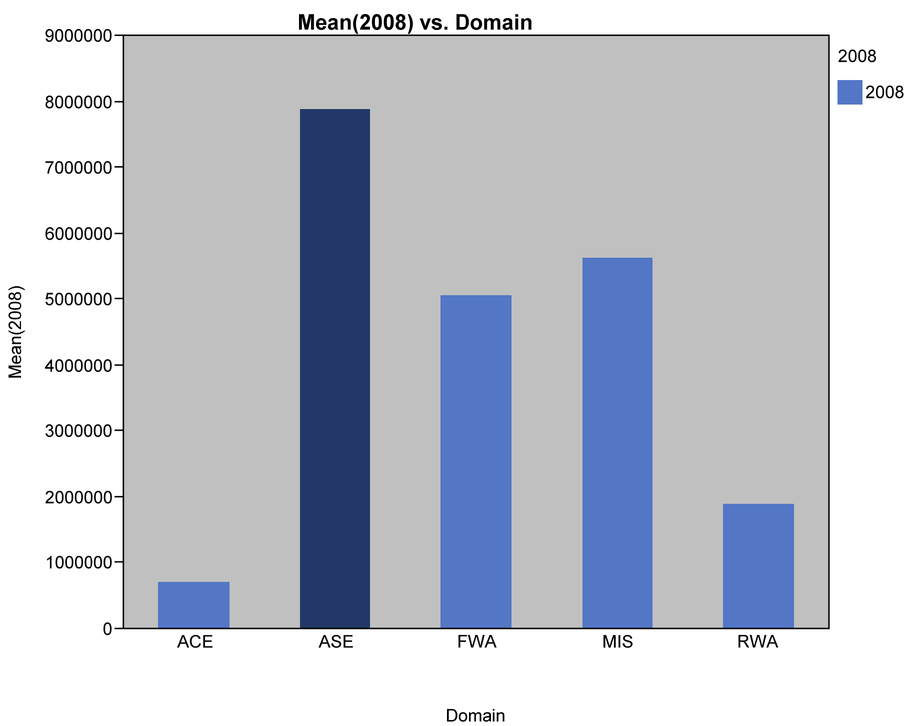


Figure 21. Mean of PRE Actual Funded Amount for FY08 by Category

Correlation analysis for these programs was computed within each category and combined when appropriate. Fixed wing aviation contained the largest amount of systems

(9) and was analyzed using FY08 PRE cost data, as shown in Figure 22. The remaining years' worth of correlations are contained in Appendix B. This analysis revealed strong correlations (greater than 0.50) between all of the variables chosen. However, the number of CSCIs within the programs exposed the most promising relationship (0.90) with FY08 funding amounts within this category.

Correlations

	FY2008 Funded Amount	Avg of Units/Systems Deployed	SUM of SLOC	Sum of CSCI/Subsystems
FY2008 Funded Amount	1.0000	0.6209	0.7509	0.9039
Avg of Units/Systems Deployed	0.6209	1.0000	0.6985	0.6566
SUM of SLOC	0.7509	0.6985	1.0000	0.7496
Sum of CSCI/Subsystems	0.9039	0.6566	0.7496	1.0000

Figure 22. Multivariate Correlations Report for PRE Data for Fixed Wing Aviation, FY08 Funded Amounts, Average Number of Systems Deployed, SLOC, and CSCIs

Next, rotary wing aviation data contained seven data points and was computed in the same manner as fixed wing using the same variable categories. The variables did not reveal the strong correlations depicted in fixed wing aviation, as shown in Figure 23. It is assumed that this occurred because of the age of the rotary aircraft. The PRE data included older aircraft that do not require a great deal of software, for example the UH–1 utility aircraft. However, the number of CSCIs and the FY08 funded amount still proved to be a significant (0.69) relationship. Additionally, the number of CSCIs compared to the total SLOC revealed a strong (0.91) relationship. The remaining years' worth of correlations are contained in Appendix B.

Correlations

	FY08 Funded Amount	Avg of Units/Systems Deployed	Total SLOC	Total CSCIs/Subsystems
FY08 Funded Amount	1.0000	0.3836	0.4241	0.6986
Avg of Units/Systems Deployed	0.3836	1.0000	-0.3781	-0.2665
Total SLOC	0.4241	-0.3781	1.0000	0.9169
Total CSCIs/Subsystems	0.6986	-0.2665	0.9169	1.0000

Figure 23. Multivariate Correlations Report for PRE Data for Rotary Wing Aviation, FY08 Funded Amounts, Average Number of Systems Deployed, SLOC, and CSCIs

The category of air combat electronics contained seven entries and was computed using the same variable categories as fixed and rotary wing aviation. The variables revealed weaker correlations between the variables and no relationship between any of the variables and the FY08 funded amounts. As anticipated, the SLOC, the total number of CSCIs/subsystems, and the average number of units/subsystems revealed strong relationships between them, as shown in Figure 24. It is worth noting that the correlation between total SLOC and the funded amount was much different than the two previous correlations. It is assumed that this difference could be attributed to fixed and rotary wing use of SLOC as a measure of their funded amounts versus air combat electronic programs, which may use another metric for requesting their maintenance funding.

Correlations

	FY08 Funded Amount	Avg of Units/Subsystems deployed	Total SLOC	Total CSCIs/Subsystems
FY08 Funded Amount	1.0000	-0.0124	-0.2517	-0.2593
Avg of Units/Subsystems deployed	-0.0124	1.0000	0.9449	0.7753
Total SLOC	-0.2517	0.9449	1.0000	0.9125
Total CSCIs/Subsystems	-0.2593	0.7753	0.9125	1.0000

Figure 24. Multivariate Correlations Report for PRE Data for Air Combat Electronics, FY08 Funded Amounts, Average Number of Systems Deployed, SLOC, and CSCIs

The category for aviation support equipment contained only two data points; therefore, the researcher determined that these points should be combined with the data for air combat electronics for analysis. The correlation was computed again with the results shown in Figure 25. By combining the two domains for the purposes of analysis, the results revealed a stronger relationship between FY08 funded amounts and CSCIs (0.76). However, this mixture decreased the relationships between SLOC, the number of deployed units, and CSCIs. Given the results of these correlations, it may not be pertinent to combine these domains for further analysis. The remaining years' worth of correlations for ACE and the combined ACE/ASE data set are contained in Appendix B.

Correlations

	FY08 Funded Amount	Avg of Units/Subsystems deployed	Total SLOC	Total CSCIs/Subsystems
FY08 Funded Amount	1.0000	-0.1213	-0.0313	0.7569
Avg of Units/Subsystems deployed	-0.1213	1.0000	0.4588	0.3980
Total SLOC	-0.0313	0.4588	1.0000	0.2895
Total CSCIs/Subsystems	0.7569	0.3980	0.2895	1.0000

Figure 25. Multivariate Correlations Report for PRE Data for Air Combat Electronics, FY08 Funded Amounts, Average Number of Systems Deployed, SLOC, and CSCIs With ASE Data

Next, the category for missile software contained three data points and was computed in the same manner as the preceding data using the same categorical variables.

The results are shown in Figure 26. Even though the number of data points was small, a strong relationship (0.88) was revealed between the FY08 funded amount and the average number of units/systems deployed. However, this data set would need to include more data points in order to be more conclusive than what is currently presented.

Correlations

	FY08 Funded Amount	Avg of Units/Systems Deployed	Total SLOC	Total CSCIs/Subsystems
FY08 Funded Amount	1.0000	0.8883	-0.0224	-0.6152
Avg of Units/Systems Deployed	0.8883	1.0000	0.4392	-0.1844
Total SLOC	-0.0224	0.4392	1.0000	0.8020
Total CSCIs/Subsystems	-0.6152	-0.1844	0.8020	1.0000

Figure 26. Multivariate Correlations Report for PRE Data for Missiles, FY08 Funded Amounts, Average Number of Systems Deployed, SLOC, and CSCIs

Finally, a combination of the fixed and rotary wing aviation data was correlated in order to determine if there were any relationships that could be revealed given that these programs all involve manned-flight platforms. This category contained 16 data points, and the results for this analysis are shown in Figure 27. By combining the data sets, the correlation analysis revealed positive relationships between the variables. In this case, the relationship between FY08 funded amounts and the number of CSCIs/subsystems contained the strongest (0.83) correlation.

Correlations

	FY08 Funded Amount	Avg of Units/Systems Deployed	Total SLOC	Total CSCI/Subsystems
FY08 Funded Amount	1.0000	0.4586	0.7763	0.8397
Avg of Units/Systems Deployed	0.4586	1.0000	0.4274	0.3395
Total SLOC	0.7763	0.4274	1.0000	0.7424
Total CSCI/Subsystems	0.8397	0.3395	0.7424	1.0000

Figure 27. Multivariate Correlations Report for PRE Data for Fixed and Rotary Wing Aviation, FY08 Funded Amounts, Average Number of Systems Deployed, SLOC, and CSCIs

However revealing these correlations were, correlation does not equal causation. Therefore, further statistical analysis was necessary in order to create a potential cost model or cost-estimating relationship. The next section uses simple linear regression analysis based on the correlation results.

B. REGRESSION ANALYSIS

1. Purpose

In order to estimate the costs associated with software maintenance, it is important to conduct regressions. This method of analysis allows a researcher to estimate the results of one variable from the input of another variable. In this case, the researcher wanted to estimate the cost (whether in actual costs, funded amounts, or labor hours) for a project's maintenance when comparing that cost to a variety of variables (SLOC counts, average number of units/subsystems deployed, number of CSCIs, etc.). In this type of analysis, it is important to regard the entire statistical package when considering accepting the regression results. For example, a researcher needs to look beyond the apparent "fit" of the data points along the regression line. While this technique provides some advantages, the next step involves examining the coefficient of determination, which explains the total variation contained within the regression equation itself and is represented by Equation 6.

$$R^2 = \frac{ExplainedVariation}{TotalVariation} = \frac{\sum\left(y_{est} - \overline{y}\right)^2}{\sum\left(y - \overline{y}\right)^2} \tag{6}$$

where y_{est} is the estimated value of y for a given value of x,

and $\overline{y}$ is the mean of our known y. (Nussbaum, 2010)

The coefficient of determination can be further explained by R^2_{adj}, which removes one degree of freedom and allows for greater variation explanation given a smaller sample size. This statistic is particularly useful considering the diminutive volume of the data sets used for this thesis. Lastly, the f test statistic was considered essential to the analysis. This test reveals whether or not the model represented by the regression equation is preferred versus having the coefficients for the dependent variables equal to zero. Typically, if the probability of calculating an f statistic is greater than 0.05, the model is considered not good, and researchers should search for an alternative. These statistics determine the strength of the regressions conducted and provide evidence for future multivariate cost models.

For the purposes of this thesis, the analysis results can be found in tables corresponding to their applicable regression graph. The criteria for designating a useable model depended on the coefficient of determination, the adjusted coefficient of determination, and the *f* statistic. Each coefficient of determination result was compared to Table 10, which allowed the researcher to conclude the utility of the model. The *f* statistic was analyzed based on whether the statistic exceeded the established 0.05 threshold. If the regression results for the *f* statistic were beyond 0.05, the researcher concluded that the dependent variable did not significantly improve the ability to predict costs (the independent variable) and, therefore, should not be used.

Table 11. Bivariate Regression Analysis Criterion

	0 –50%	**51–60%**	**61–70%**	**71–80%**	**81–99%**
Coefficient of Determination	Weak	Inconclusive	Moderately strong	Strong	Very strong
Coefficient of Determination (adjusted)	Weak	Inconclusive	Moderately strong	Strong	Very strong

2. Warner Robins and ISPAN

The bivariate regressions executed on this data set attempted to determine the possible variables that could be used in a best fit model. The correlations demonstrated that total SLOC and the percentage of effort in perfective maintenance could result as a candidate best fit model. Therefore, the first regression placed the total SLOC as the dependent variable and the percentage of effort in perfective maintenance as the independent. The results of this analysis are shown in Figures 28 and 29.

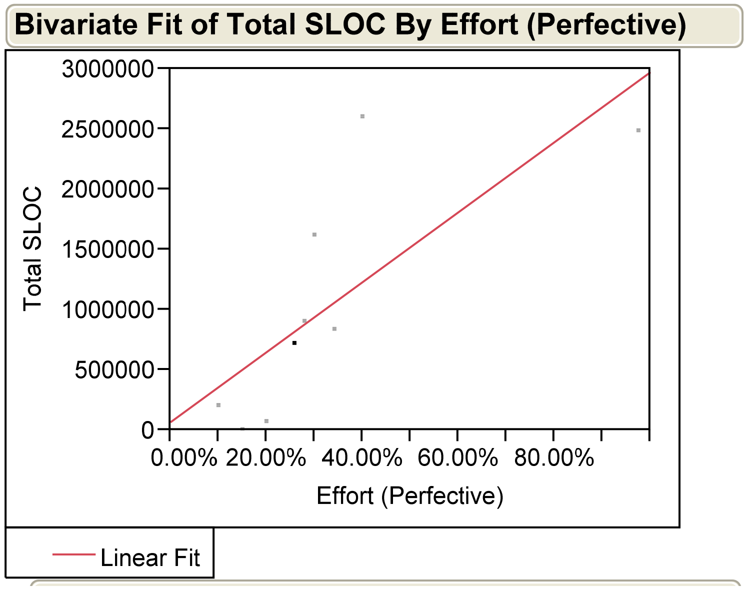


Figure 28. Linear Fit Regression for SLOC and Percentage of Effort in Perfective Maintenance

The linear relationship equation for Figure 28 is represented by Equation 7.

$$\text{Total SLOC} = 65314.3 + 2909117 * \text{Effort (Perfective)} \tag{7}$$

Table 12. Bivariate Regression Results

	Results	Researcher's Interpretation
Coefficient of Determination (R^2)	57%	Inconclusive
Adjusted Coefficient of Determination (R^2_{adj})	51%	Inconclusive
***f* statistic**	0.0185	Good

Only slightly more than 50% of this model's variability could be explained through the coefficients of determination. Additionally, the *f* value (0.0185) did not surpass the threshold of 0.05, which implies that this could be a model candidate if there are no superior alternatives.

Summary of Fit

RSquare	0.571231
RSquare Adj	0.509978
Root Mean Square Error	684622.1
Mean of Response	1053121
Observations (or Sum Wgts)	9

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	1	4.3711e+12	4.371e+12	9.3258
Error	7	3.281e+12	4.687e+11	**Prob > F**
C. Total	8	7.652e+12		0.0185*

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	65314.368	395864.9	0.16	0.8736
Effort (Perfective)	2909117	952616.7	3.05	0.0185*

Figure 29. Whole Model Statistical Tables for SLOC and Percentage of Effort in Perfective Maintenance

3. Picatinny Arsenal

Simple bivariate regressions were executed on the data sets in order to determine the best variables for inclusion in a best fit model. The Picatinny Arsenal correlations revealed that the overall cost category contained a strong relationship with the number of New SLOC (added) in the maintenance. Therefore, the first regression placed overall costs as the dependent variable and SLOC New (added) as the independent. The results of this analysis are shown in Figures 30 and 31.

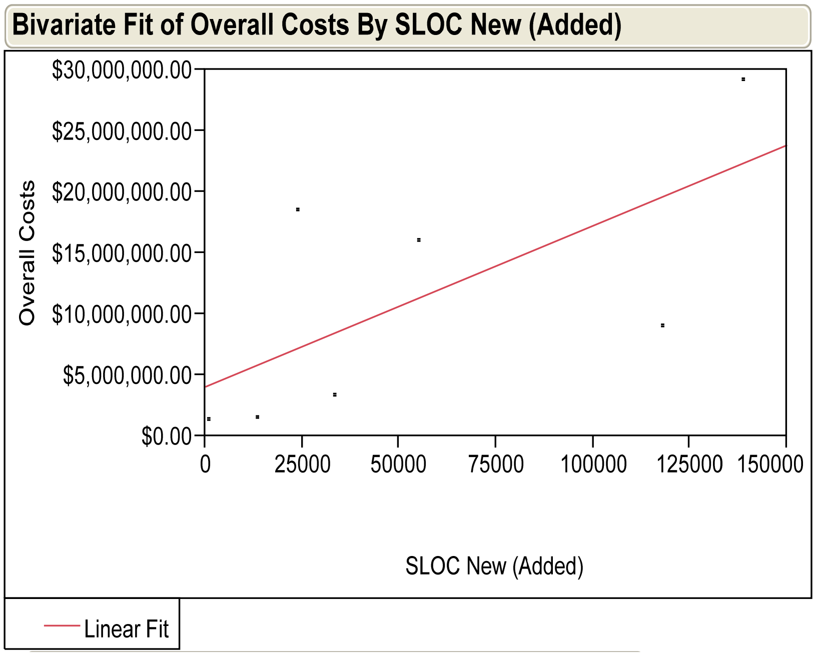


Figure 30. Linear Fit Regression for Overall Costs and SLOC New (Added)

The linear relationship equation for Figure 30 is represented by Equation 8.

$$\text{Overall Costs} = 4048176.7 + 132.0 * \text{SLOC New (Added)} \qquad (8)$$

Table 13. Bivariate Regression Results

	Results	**Researcher's Interpretation**
Coefficient of Determination (R^2)	47%	Weak
Adjusted Coefficient of Determination (R^2_{adj})	34%	Weak
***f* statistic**	0.098	Not Good

Less than 50% of this model's variability could be explained through the coefficients of determination. Additionally, the *f* value (0.098) surpassed the threshold of 0.05, which implies that this is not a good model to use.

Summary of Fit

RSquare	0.450561
RSquare Adj	0.340673
Root Mean Square Error	8522546
Mean of Response	11271330
Observations (or Sum Wgts)	7

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	1	2.9781e+14	2.978e+14	4.1002
Error	5	3.6317e+14	7.263e+13	**Prob > F**
C. Total	6	6.6098e+14		0.0988

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	4048176.7	4806350	0.84	0.4381
SLOC New (Added)	132.03274	65.20478	2.02	0.0988

Figure 31. Whole Model Statistical Tables for Overall Costs and SLOC New (Added)

Another simple regression was performed using total effort (in man-months) against SLOC New (Added) since this was determined to possess a strong relationship during correlation analysis. The results are shown in Figures 32 and 33.

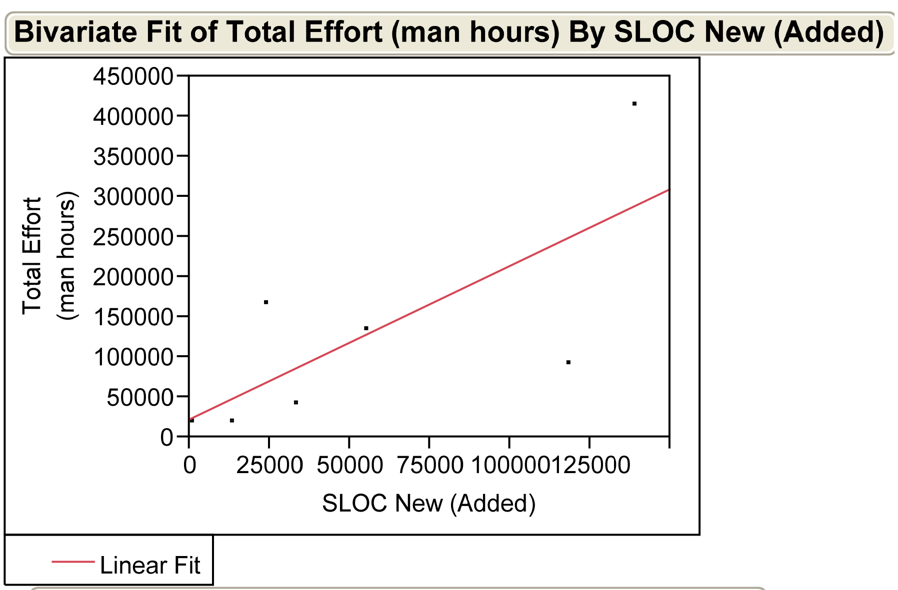


Figure 32. Linear Fit Regression for Total Effort and SLOC New (Added)

The linear relationship equation for Figure 32 is represented by Equation 9.

$$\text{Total Effort} = 22717.3 + 1.9 * \text{SLOC New (Added)} \tag{9}$$

Table 14. Bivariate Regression Results

	Results	Researcher's Interpretation
Coefficient of Determination (R^2)	54%	Inconclusive
Adjusted Coefficient of Determination (R^2_{adj})	44%	Weak
***f* statistic**	0.059	Not Good

Less than 50% of this model's variability could be explained through the coefficients of determination. Additionally, the *f* value (0.059) surpassed the threshold of 0.05, which implies that this is not a good model to use. Based on the data from these regressions, it would be difficult to derive an effective model for cost prediction based on the results.

Summary of Fit

RSquare	0.541625
RSquare Adj	0.44995
Root Mean Square Error	102965
Mean of Response	127470.9
Observations (or Sum Wgts)	7

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	1	6.2636e+10	6.264e+10	5.9081
Error	5	5.3009e+10	1.06e+10	**Prob > F**
C. Total	6	1.1565e+11		0.0593

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	22717.337	58067.84	0.39	0.7117
SLOC New (Added)	1.9148002	0.78777	2.43	0.0593

Figure 33. Whole Model Statistical Tables for Total Effort and SLOC New (Added)

4. Integrated Strategic Planning and Analysis Network

Similar to the Picatinny data, the ISPAN data was subjected to regression tests in order to determine the best variables for inclusion in a best fit model. The ISPAN data correlations computed that the FTE maintenance category contained a strong relationship with the number of SLOC in the software. Therefore, the first regression analyzed FY08 data and placed FTE maintenance as the dependent variable with SLOC as the independent. This analysis included all six ISPAN programs. The results are shown in Figures 34 and 35.

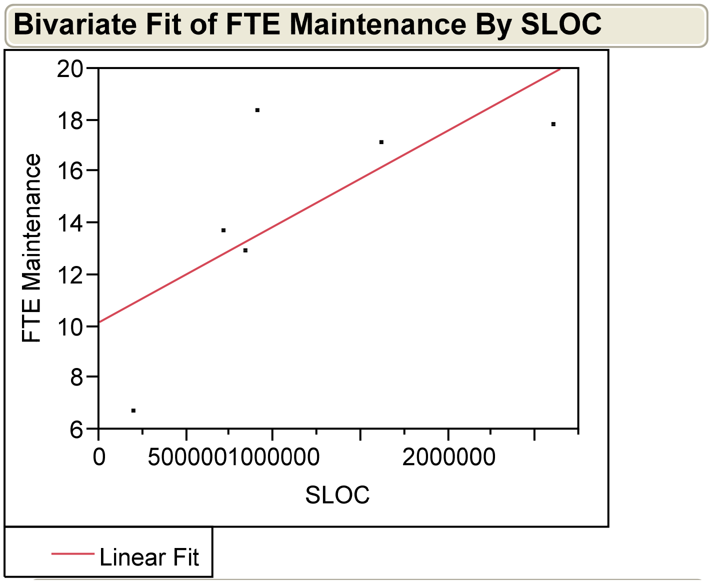


Figure 34. Linear Fit Regression for FTE Maintenance and SLOC for Six ISPAN Programs

The linear relationship equation for Figure 34 is represented by Equation 10.

$$\text{FTE maintenance} = 10.1 + 3.7\text{e-}6 * \text{SLOC} \tag{10}$$

Table 15. Bivariate Regression Results

	Results	**Researcher's Interpretation**
Coefficient of Determination (R^2)	51%	Inconclusive
Adjusted Coefficient of Determination (R^2_{adj})	38%	Weak
***f* statistic**	0.11	Not Good

Less than 50% of this model's variability could be explained through the coefficients of determination. Additionally, the *f* value (0.11) surpassed the threshold of 0.05, which implies that this is not a good model to use.

Summary of Fit

RSquare	0.510947
RSquare Adj	0.388683
Root Mean Square Error	3.44152
Mean of Response	14.43333
Observations (or Sum Wgts)	6

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	1	49.497091	49.4971	4.1791
Error	4	47.376243	11.8441	**Prob > F**
C. Total	5	96.873333		0.1104

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	10.176741	2.511886	4.05	0.0155*
SLOC	3.7211e-6	1.82e-6	2.04	0.1104

Figure 35. Whole Model Statistical Tables for FTE Maintenance and SLOC for Six ISPAN Programs

Another regression was executed using FTE maintenance against defects since this was determined to possess a strong relationship during correlation analysis. However, as was done during correlation analysis, the Theater Integrated Planning System was removed. The results are shown in Figures 36 and 37.

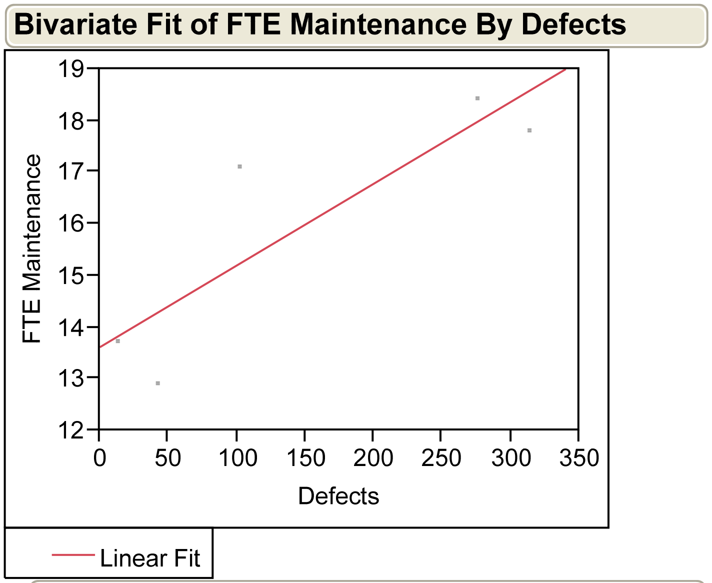


Figure 36. Linear Fit Regression for FTE Maintenance and Defects for Five ISPAN Programs

The linear relationship equation for Figure 36 is represented by Equation 11.

$$\text{FTE maintenance} = 13.6 + 0.015 * \text{Defects} \qquad (11)$$

Table 16. Bivariate Regression Results

	Results	**Researcher's Interpretation**
Coefficient of Determination (R^2)	75%	Strong
Adjusted Coefficient of Determination (R^2_{adj})	66%	Moderately strong
***f* statistic**	0.057	Not Good

Only slightly more than 50% of this model's variability could be explained through the coefficients of determination. Additionally, the *f* value (0.057) barely surpassed the threshold of 0.05, which implies that this could be a model candidate if there are no superior alternatives.

Summary of Fit

RSquare	0.750506
RSquare Adj	0.667341
Root Mean Square Error	1.445026
Mean of Response	15.98
Observations (or Sum Wgts)	5

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	1	18.843701	18.8437	9.0243
Error	3	6.264299	2.0881	**Prob > F**
C. Total	4	25.108000		0.0575

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	13.612534	1.01917	13.36	0.0009*
Defects	0.0158465	0.005275	3.00	0.0575

Figure 37. Whole Model Statistical Tables for FTE Maintenance and SLOC for Five ISPAN Programs

5. Lockheed Martin Systems Integration

In order to determine the variables for a best fit model, the Lockheed Martin data was subjected to regression tests. The Lockheed Martin data correlations computed that the labor month's category contained the strongest relationship with the amount of new code and a weaker relationship with the amount of modified code and the total defects in the software. Therefore, the first regression analyzed placed labor months as the dependent variable and the amount of new code as the independent variable. This analysis excluded two programs that reported zero modified code. The results are shown in Figures 38 and 39.

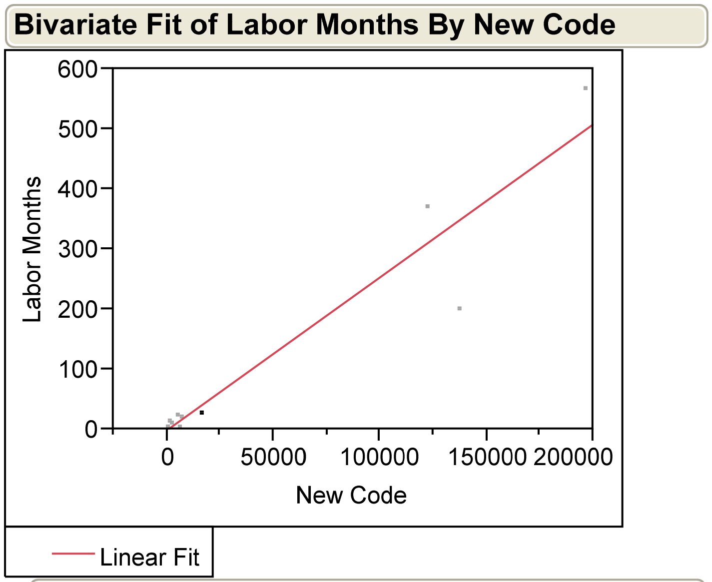


Figure 38. Linear Fit Regression for Labor Months and New Code for Fourteen Lockheed Martin Programs

The linear relationship equation for Figure 38 is represented by Equation 12.

$$\text{Labor Months} = -1.015 + 0.0025 * \text{New Code} \tag{12}$$

Table 17. Bivariate Regression Results

	Results	**Researcher's Interpretation**
Coefficient of Determination (R^2)	92%	Strong
Adjusted Coefficient of Determination (R^2_{adj})	91%	Strong
***f* statistic**	<0.0001	Good

More than 90% of this model's variability could be explained through the coefficients of determination. Additionally, the *f* value (<0.0001) did not surpass the threshold of 0.05, which implies that this could be a model candidate to use.

Summary of Fit

RSquare	0.918737
RSquare Adj	0.911965
Root Mean Square Error	51.36636
Mean of Response	89.28429
Observations (or Sum Wgts)	14

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	1	357961.07	357961	135.6683
Error	12	31662.03	2639	Prob > F
C. Total	13	389623.11		<.0001*

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	-1.015442	15.76602	-0.06	0.9497
New Code	0.0025467	0.000219	11.65	<.0001*

Figure 39. Whole Model Statistical Tables for Labor Months and New Code for Fourteen Lockheed Martin Programs

A second regression was executed using labor months against the amount of modified code since this was determined to possess a strong relationship during correlation analysis. However, contrary to the correlation analysis, two programs that contained zero modified code were removed. The results are shown in Figures 40 and 41.

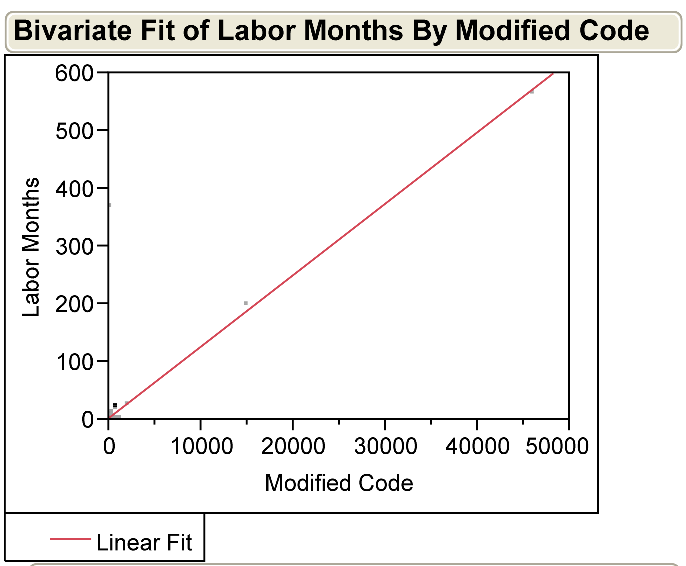


Figure 40. Linear Fit Regression for Labor Months and Modified Code for Twelve Lockheed Martin Programs

The linear relationship equation for Figure 40 is represented by Equation 13.

$$\text{Labor Months} = 3.45 + 0.012 * \text{Modified Code} \tag{13}$$

Table 18. Bivariate Regression Results

	Results	**Researcher's Interpretation**
Coefficient of Determination (R^2)	99%	Strong
Adjusted Coefficient of Determination (R^2_{adj})	99%	Strong
***f* statistic**	<0.0001	Good

More than 90% of this model's variability could be explained through the coefficients of determination. Additionally, the *f* value (<.00001) did not surpass the threshold of 0.05, which implies that this could be a model candidate to use.

Summary of Fit

RSquare	0.997298
RSquare Adj	0.997028
Root Mean Square Error	9.004868
Mean of Response	73.13583
Observations (or Sum Wgts)	12

Analysis of Variance

Source	**DF**	**Sum of Squares**	**Mean Square**	**F Ratio**
Model	1	299324.08	299324	3691.365
Error	10	810.88	81	**Prob > F**
C. Total	11	300134.95		<.0001*

Parameter Estimates

Term	**Estimate**	**Std Error**	**t Ratio**	**Prob>|t|**
Intercept	3.4587149	2.841216	1.22	0.2514
Modified Code	0.0123795	0.000204	60.76	<.0001*

Figure 41. Whole Model Statistical Tables for Labor Months and Modified Code for Twelve Lockheed Martin Programs

A third regression was executed using labor months against the amount of defects in the software since this was also determined to possess a strong relationship during correlation analysis. However, contrary to the correlation analysis, one program reported zero defects, so that program was removed for this analysis. The results are shown in Figures 42 and 43.

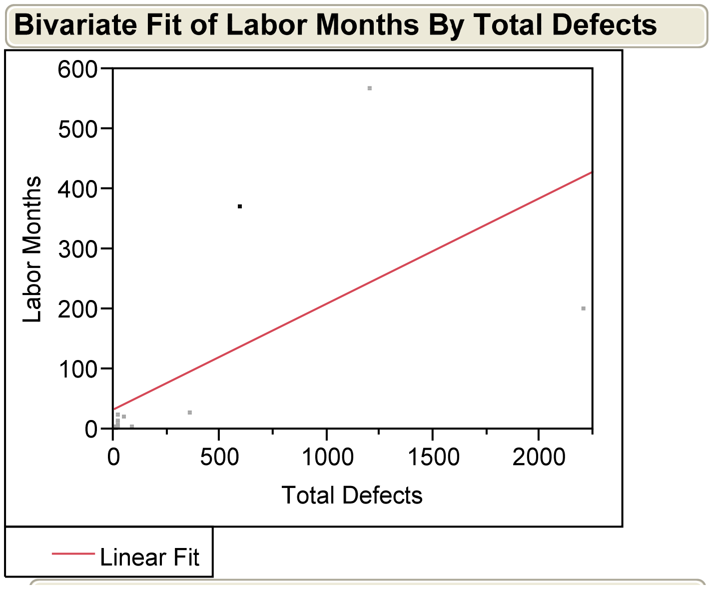


Figure 42. Linear Fit Regression for Labor Months and Defects for Twelve Lockheed Martin Programs

The linear relationship equation for Figure 42 is represented by Equation 14.

$$\text{Labor Months} = 33.9 + 0.17 * \text{Total Defects} \tag{14}$$

Table 19. Bivariate Regression Results

	Results	**Researcher's Interpretation**
Coefficient of Determination (R^2)	41%	Weak
Adjusted Coefficient of Determination (R^2_{adj})	36%	Weak
***f* statistic**	0.01	Good

Less than 40% of this model's variability could be explained through the coefficients of determination. Additionally, the *f* value (0.01) did not surpass the threshold of 0.05, which implies that this model may be useful if there are no other alternatives.

Summary of Fit

RSquare	0.419563
RSquare Adj	0.366796
Root Mean Square Error	141.8795
Mean of Response	95.97154
Observations (or Sum Wgts)	13

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	1	160056.57	160057	7.9512
Error	11	221427.61	20130	**Prob > F**
C. Total	12	381484.18		0.0167*

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	33.915556	45.0862	0.75	0.4677
Total Defects	0.175911	0.062384	2.82	0.0167*

Figure 43. Whole Model Statistical Tables for Labor Months and Total Defects for Thirteen Lockheed Martin Programs

6. NAVAIR PRE Data

The NAVAIR PRE data correlations revealed that the FY08 funded amount category contained a number of strong relationships with variables from the data provided. Bivariate regressions were calculated for each category according to the strength of the correlation. Those correlations that disclosed the highest positive correlation were used to populate the regression.

The fixed wing aviation correlations revealed that the FY08 funded amount category contained a strong relationship with the sum of CSCIs/subsystems associated with the program. Therefore, the first regression placed the FY08 funded amount as the dependent variable and the sum of CSCIs/subsystems as the independent variable. The results of this analysis are shown in Figures 44 and 45.

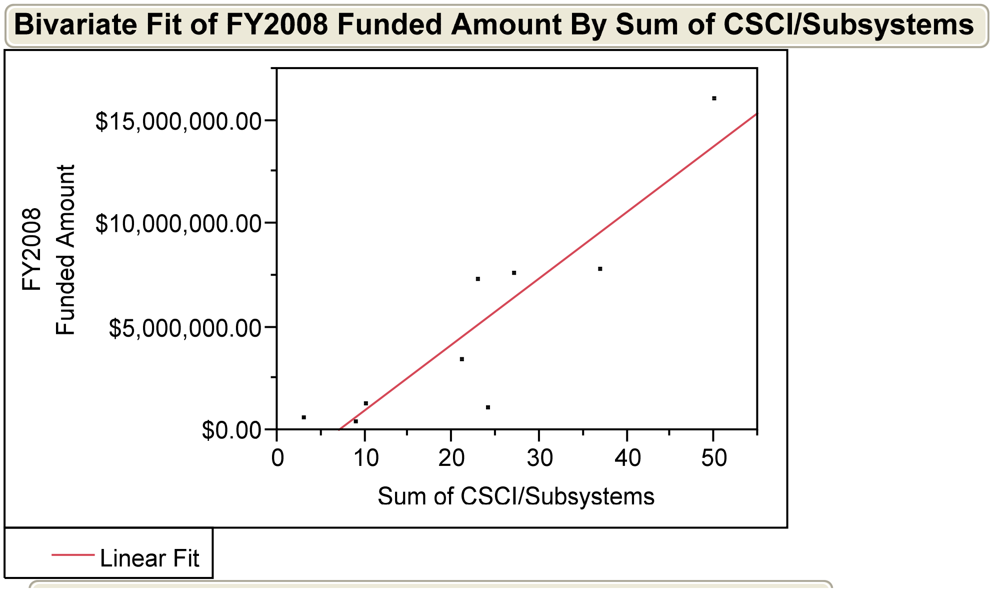


Figure 44. Linear Fit Regression for FY08 Funded Amount and Sum of CSCIs/Subsystems for Ten Fixed Wing Aviation Programs

The linear relationship equation for Figure 44 is represented by Equation 15.

$$\text{FY08 Funded Amount} = -2{,}208{,}978 + 319866.6 * \text{Sum of CSCIs/Subsystems} \quad (15)$$

Table 20. Bivariate Regression Results

	Results	Researcher's Interpretation
Coefficient of Determination (R^2)	81%	Very strong
Adjusted Coefficient of Determination (R^2_{adj})	79%	Moderately strong
***f* statistic**	0.0008	Good

Eighty percent of this model's variability could be explained through the coefficients of determination. Additionally, the *f* value (0.0008) did not surpass the threshold of 0.05, which implies that this could be a model candidate to use.

Summary of Fit	
RSquare	0.817078
RSquare Adj	0.790946
Root Mean Square Error	2365483
Mean of Response	5041333
Observations (or Sum Wgts)	9

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	1	1.7496e+14	1.75e+14	31.2676
Error	7	3.9169e+13	5.596e+12	**Prob > F**
C. Total	8	2.1413e+14		0.0008*

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	-2208978	1517538	-1.46	0.1888
Sum of CSCI/Subsystems	319866.67	57203.39	5.59	0.0008*

Figure 45. Whole Model Statistical Tables for FY08 and Sum of CSCIs/Subsystems for Nine Fixed Wing Aviation Programs

The rotary wing aviation correlations revealed that the FY08 funded amount category contained a strong relationship with the sum of CSCIs/subsystems associated with the program. Therefore, the first regression placed the FY08 funded amount as the dependent variable and the sum of CSCIs/subsystems as the independent variable. The results of this analysis are shown in Figures 46 and 47.

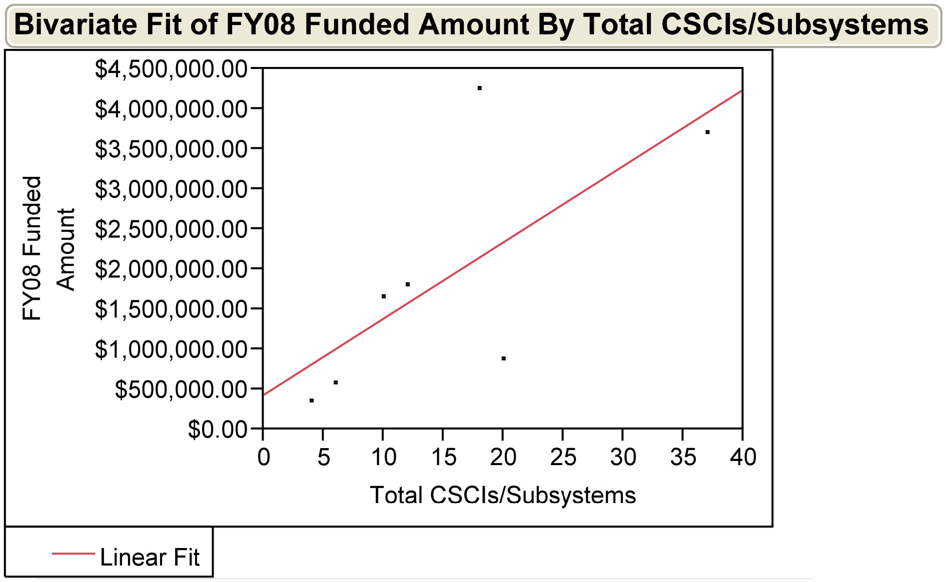


Figure 46. Linear Fit Regression for FY08 Funded Amount and Sum of CSCIs/Subsystems for Seven Rotary Wing Aviation Programs

The linear relationship equation for Figure 46 is represented by Equation 16.

FY08 Funded Amount = 432,009.5 + 95298.4.6 * Total of CSCIs/Subsystems (16)

Table 21. Bivariate Regression Results

	Results	**Researcher's Interpretation**
Coefficient of Determination (R^2)	48%	Inconclusive
Adjusted Coefficient of Determination (R^2_{adj})	39%	Inconclusive
***f* statistic**	0.08	Not Good

Only slightly more than 40% of this model's variability could be explained through the coefficients of determination. Additionally, the *f* value (0.08) surpassed the threshold of 0.05, which implies that this is not a good model to use.

Summary of Fit

RSquare	0.488004
RSquare Adj	0.385605
Root Mean Square Error	1198238
Mean of Response	1888714
Observations (or Sum Wgts)	7

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	1	6.8425e+12	6.842e+12	4.7657
Error	5	7.1789e+12	1.436e+12	**Prob > F**
C. Total	6	1.4021e+13		0.0808

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	432009.48	806457	0.54	0.6151
Total CSCIs/Subsystems	95298.445	43653.81	2.18	0.0808

Figure 47. Whole Model Statistical Tables for FY08 and Sum of CSCIs/Subsystems for Seven Fixed Wing Aviation Programs

The air combat electronics correlations revealed that there were no positive correlations between the FY08 funded amount category and any of the potential independent variables associated with the program. Therefore, there were no regressions calculated on this data. However, when the ACE data was combined with the aviation support equipment, the FY08 funded amount category contained a strong relationship with the sum of CSCIs/subsystems associated with the program. Therefore, this regression analysis placed the FY08 funded amount as the dependent variable and the sum of CSCIs/subsystems as the independent variable. The results of this analysis are shown in Figures 48 and 49.

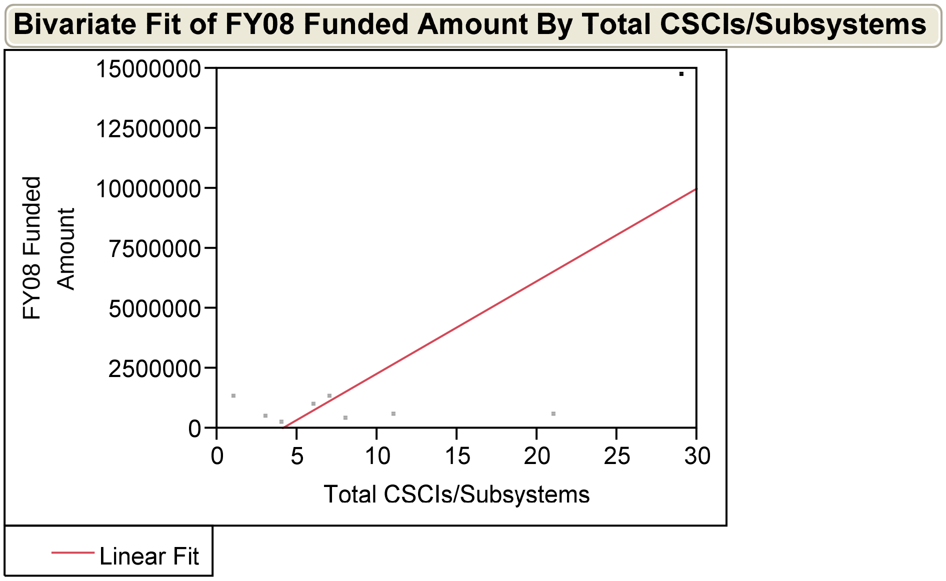


Figure 48. Linear Fit Regression for FY08 Funded Amount and Sum of CSCIs/Subsystems for Seven ACE and Two ASE Programs

The linear relationship equation for Figure 48 is represented by Equation 17.

$$\text{FY08 Funded Amount} = -1553693 + 385902.6 * \text{Total CSCIs/Subsystems} \qquad (17)$$

Table 22. Bivariate Regression Results

	Results	**Researcher's Interpretation**
Coefficient of Determination (R^2)	57%	Inconclusive
Adjusted Coefficient of Determination (R^2_{adj})	51%	Inconclusive
***f* statistic**	0.01	Good

Only slightly more than 50% of this model's variability could be explained through the coefficients of determination. Additionally, the *f* value (0.01) did not surpass the threshold of 0.05, which implies that this could be a model candidate if there are no superior alternatives.

Summary of Fit	
RSquare	0.572877
RSquare Adj	0.511859
Root Mean Square Error	3279363
Mean of Response	2305333
Observations (or Sum Wgts)	9

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	1	1.0097e+14	1.01e+14	9.3887
Error	7	7.528e+13	1.075e+13	**Prob > F**
C. Total	8	1.7625e+14		0.0182*

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	-1553693	1667658	-0.93	0.3825
Total CSCIs/Subsystems	385902.65	125943.2	3.06	0.0182*

Figure 49. Whole Model Statistical Tables for FY08 and Sum of CSCIs/Subsystems for Seven ACE and Two ASE Programs

The missile category correlations revealed that the FY08 funded amount category contained a strong relationship with the average of units/systems associated with the program. However, there were only three programs to analyze. Nevertheless, these systems were subjected to regression analysis in order to discover any possible useful information. The regression placed the FY08 funded amount as the dependent variable and the average of units/systems as the independent variable. The results of this analysis are shown in Figures 50 and 51.

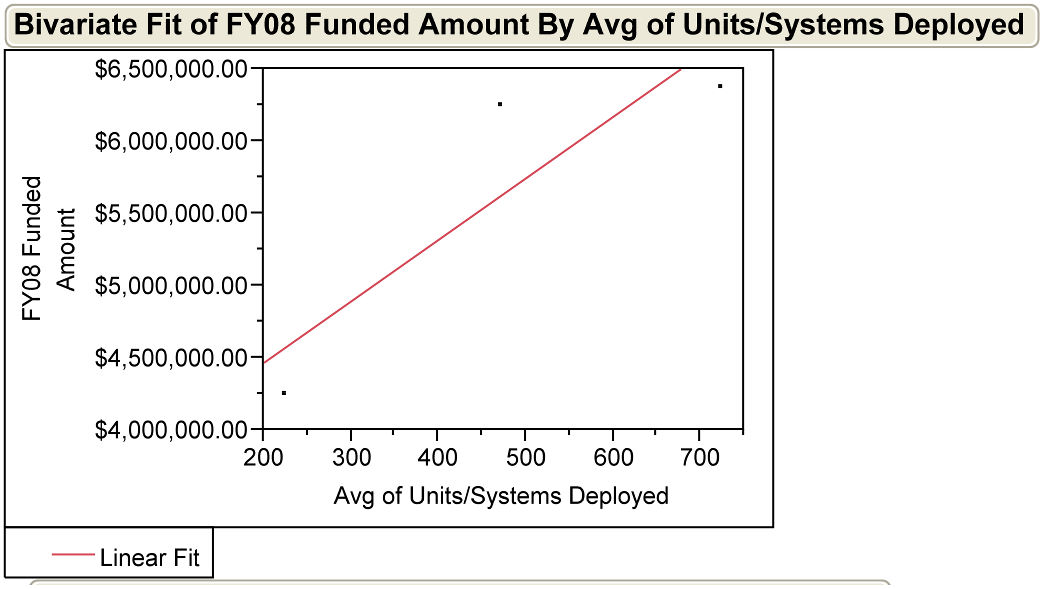


Figure 50. Linear Fit Regression for FY08 Funded Amount and Average of Units/Systems for Three Missile Programs

The linear relationship equation for Figure 50 is represented by Equation 18.

$$\text{FY08 Funded Amount} = 432{,}009.5 + 95298.46 * \text{Total of CSCIs/Subsystems} \quad (18)$$

Table 23. Bivariate Regression Results

	Results	**Researcher's Interpretation**
Coefficient of Determination (R^2)	78%	Strong
Adjusted Coefficient of Determination (R^2_{adj})	57%	Inconclusive
***f* statistic**	0.303	Not Good

More than 60% of this model's variability could be explained through the coefficients of determination. Additionally, the *f* value (0.303) surpassed the threshold of 0.05, which implies that this is not a good model to use.

Summary of Fit

RSquare	0.789098
RSquare Adj	0.578195
Root Mean Square Error	775697.8
Mean of Response	5625333
Observations (or Sum Wgts)	3

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	1	2.2513e+12	2.251e+12	3.7415
Error	1	6.0171e+11	6.017e+11	**Prob > F**
C. Total	2	2.853e+12		0.3038

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	3612782.8	1132744	3.19	0.1934
Avg of Units/Systems Deployed	4260.8692	2202.792	1.93	0.3038

Figure 51. Whole Model Statistical Tables for FY08 and Average of Units/Systems for Three Missile Programs

The fixed and rotary wing aviation combination correlations revealed that the FY08 funded amount category contained a strong relationship with the sum of CSCIs/subsystems associated with the programs. Therefore, the regression placed the FY08 funded amount as the dependent variable and the sum of CSCIs/subsystems as the independent variable. The results of this analysis are shown in Figures 52 and 53.

Bivariate Fit of FY08 Funded Amount By Total CSCI/Subsystems

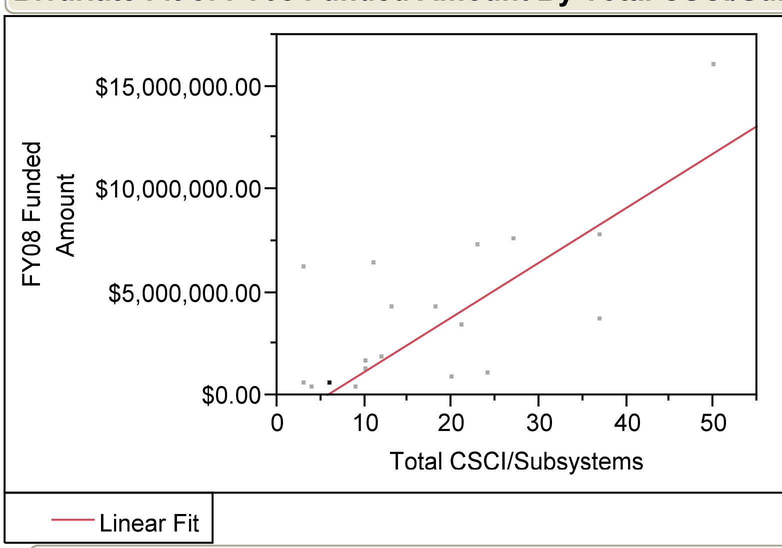


Figure 52. Linear Fit Regression for FY08 Funded Amount and Sum of CSCIs/Subsystems for a Combination of Fixed and Rotary Wing Programs

The linear relationship equation for Figure 52 is represented by Equation 19.

$$\text{FY08 Funded Amount} = -1{,}494{,}262 + 265277.1 * \text{Total of CSCIs/Subsystems} \quad (19)$$

Table 24. Bivariate Regression Results

	Results	**Researcher's Interpretation**
Coefficient of Determination (R^2)	71%	Strong
Adjusted Coefficient of Determination (R^2_{adj})	68%	Moderately Strong
***f* statistic**	<0.0001	Good

More than 69% of this model's variability could be explained through the coefficients of determination. Additionally, the *f* value (<0.0001) did not surpass the threshold of 0.05, which implies that this could be a model candidate to use.

Summary of Fit

RSquare	0.705065
RSquare Adj	0.683998
Root Mean Square Error	2372930
Mean of Response	3662063
Observations (or Sum Wgts)	16

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	1	1.8845e+14	1.885e+14	33.4680
Error	14	7.8831e+13	5.631e+12	**Prob > F**
C. Total	15	2.6728e+14		<.0001*

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	-1494262	1070675	-1.40	0.1846
Total CSCI/Subsystems	265277.13	45854.8	5.79	<.0001*

Figure 53. Whole Model Statistical Tables for FY08 and Sum of CSCIs/Subsystems for a Combination of Fixed and Rotary Wing Programs

7. Summary

The data sets provided for this thesis were from diverse sources, and the associated analyses revealed this disparate nature. The correlations validated several of the researcher's assumptions, including the assumption that the more SLOC to maintain, the higher the hours spent maintaining the code. However, this analysis also questioned the researcher's supposition about software reuse and disclosed that the amount of code reuse does not relate to the amount of cost or effort. Additionally, the discovery of a relationship between subsystems/CSCIs and costs was exposed.

The regression analysis proved to be the most enlightening task of this thesis. Based on the data, the results demonstrated that using SLOC counts to estimate costs proved to be an inconsistent method, unless the code was categorized by modified and new. The PRE data uncovered the notion of the number of subsystems/CSCIs and their relationship with funded amounts. This was particularly interesting since the number of CSCIs could reveal the complexity of the software and the maintenance challenges. Lastly, the number of defects reported also showed that this variable could be useful in a model, if calculated with additional software attributes.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSIONS AND RECOMMENDATIONS

A. SUMMARY OF FINDINGS

The diverse nature of the data provided for this thesis constrained the researcher's ability to create a model for the cost of software maintenance. However, there were a number of findings that may assist a program manager to estimate the cost of the software associated with a program. More important, these findings highlight the need for better reporting from those sources of software maintenance support in order to build more accurate models in the future.

The first observation is that the traditional total amount of SLOC metric does not accurately reflect the amount of effort required to maintain the software unless categorized by the type of SLOC maintained. A strong correlation between the total amount of SLOC and costs (whether they are actual costs, labor months, or FTE work) could not be determined. None of the bivariate models created supported using total SLOC as a sole factor for determining costs. However, SLOC is one of the major inputs to any of the software cost-estimation models employed. This analysis supports the use of additional information beyond the more easily attained total SLOC count as a method to estimate software maintenance.

The next observation is that the number of defects reported would be an accurate measure of the costs for post-production support. Strong relationships were derived between the designated cost category and the reported number of defects from the correlation analysis and the regressions executed on two programs. Additionally, the regressions that included defect counts were proven to be useful. Unfortunately, this data is dependent upon where the software is during development. If defects are reported during the testing phase of development, this information may be useful to a program manager to estimate future maintenance costs. However, the best defect data is still going to be derived from software currently in service.

The third observation is that the number of CSCIs was discovered to be highly correlated with the actual funded amount from NAVAIR's PRE data. The regressions computed revealed that the number of CSCIs/subsystems did provide useable models (more so than other data) for estimating the maintenance costs for those particular programs. This conclusion does not indicate that the number of CSCIs/subsystems associated with a program will provide accurate costs for maintenance. It does imply that the number of CSCIs/subsystems associated with a program could disclose the complexity of the software, which may well correlate to the maintenance costs if more information regarding the CSCIs/subsystems is provided. This information may provide program managers with a better understanding of the cost drivers in software maintenance.

The final observation is that the information reported by various contractors and government agencies does not provide enough detail to permit the creation of a robust software maintenance estimation cost model. As evidenced by the disparate amount of data collected, many data collection systems used by maintainers record their efforts and the particulars of whatever software they are tasked to support. However, more standardization is required across the software maintenance community in order to ensure that the data being recorded can be employed beyond the agency or contractor.

B. SPECIFIC RECOMMENDATIONS

Currently, the software resources data report (SRDR) retained by the Defense Cost and Resource Center (DCARC) requires developers to report information related to software development and upgrade costs. These reports can be done by contractors, government design activities, or a mixture of both (DoD, 2004). The reports require the submission of a DD Form 2630-2 to the DCARC within 60 days of the project start. The initial developer report provides an estimate of the work about to be performed. The final developer report (DD Form 2630-3), which reports actual data concerning the software, is then submitted to the DCARC within 60 days of delivery. This information is captured in the DCARC database and is available to those with a need to know.

A similar method of software maintenance needs to be implemented that would permit the capture of actual resources used to complete maintenance. Once the information is submitted to the appropriate Service's Visibility and Management of

Operating and Support Costs (VAMOSC) center, it would be categorized by application domain (aviation, ships, ground weapons, command and control platforms, etc.) for easy access, dependent upon the user's desires. The required information to populate the report would be programming languages (which relates to program complexity), number of subsystems/CSCIs, defect counts and their type, labor hours charged toward the maintenance provided, and SLOC by category (base, reuse, new, and modified). In order to be sensitive to contractor proprietary concerns, it would not be necessary to report labor rates or actual billing amounts. The labor effort would be reported by maintenance performed (corrective, perfective, or adaptive) in man-hours. This information could then be used as a basis for program managers to build and design their own estimation models.

C. FUTURE RESEARCH

Estimating the cost of software maintenance is a challenging problem for a variety of reasons. Many practitioners continue to postulate the factors that comprise software maintenance. Even more experts debate which costs can be (and should be) attributed to software support. Therefore, any research that attempts to contribute to this subject's body of knowledge should be regarded as pioneering work and used for further exploration. Due to recent budgetary concerns, the field should garner a great deal of attention. Therefore, the maneuver space available to the next researcher is dependent only on the determination of the researcher and the availability of the data.

This thesis described the current software maintenance cost-estimation models in use by the acquisition community. A researcher could examine these models to determine their accuracy in light of actual maintenance costs. This may prove difficult, considering that SLIM and SEER–SEM are commercial products. However, the researcher may be able to obtain the data provided to these companies and gauge their effectiveness. The case could then be made for whether it is worth the investment to use these products versus an open-source cost-estimation product like COCOMO II.

This thesis collected as much information as possible from a variety of sources across several application domains. Future research could examine one particular domain, narrow the scope to one program with several years' worth (at least five) of software maintenance, and build a predictive cost model for that one system. This effort would contribute to the data collection efforts for at least one system that could then be used by other similar systems as an estimating tool while they are still in development.

APPENDIX A.

DATA ITEM DESCRIPTION SOFTWARE MAINTENANCE DATA COLLECTION (VERSION 1.3)

Title: Software Maintenance Data Collection

Use/Relationship: This Data Item Description (DID) identifies and describes the data being collected to build a software operations and maintenance cost database. This Software Maintenance Data Collection form is not a management or measurement report. It is not intended for tracking progress, nor does it intend to collect financial information. Rather, its purpose is to collect empirical data during software operations and maintenance for use in developing benchmarks and estimating relationships, and calibrating models. These data will also be used to substantiate budgets used for future maintenance appropriations. The accompanying Excel form is provided for ease of data entry.

Timing: Because we are collecting both estimates and actuals for many of the measures identified, the best time to capture data is at the start and end of a cycle. For example, size in source lines of code would be captured as an estimate at the beginning of a release and the end with a measurement of the actuals, which can be accomplished with a code counter such as the University of Southern California (USC) Unified Code Counter (UCC), measuring actual size and the number of lines added, deleted, changed, and reused from version to version (using the tool's differential counting capability).

Additionally, data needs to be captured on an annual basis when releases are multi-year because that is how budgets are allocated. For multi-year projects, the estimate data must therefore be collected at the start of the cycle, updated with a cost and schedule to complete the start of the next fiscal year, and finalized with actuals when the release is provided to the field. Conversely, when there are several releases during a fiscal year, data snapshots are needed at the beginning and end of each release.

Information Needs:

The following data items should be collected for entry into the maintenance cost and quality database as a record for each project version released to the field. Those data items identified as **"Mandatory"** represent the minimum data set to be collected. Such data includes both contextual as well as measured values. Data are desired in as raw a form as possible (e.g., effort in hours as a direct output from the timecard system) so that any normalization steps may be traced and validated.

Indentifying Information (Mandatory)

A description of the project and associated software development process provides vital context for the subsequent data to be collected. In aggregate data analysis, all identifying information will be stripped so that each individual data point remains "anonymous."

- **Organization (contracted or in-house)–**Identify whether the version or release was done in-house by a government and contractor team or was contracted externally. If internal, provide the name of the responsible life cycle support center. If contracted, provide the name(s) of the contractor(s). Be sure to include all subcontractors in order to provide a complete accounting of the effort.

- **Program Name–**The name of the program under which the effort is being accomplished.
- **System Name**–The name of the system of which the software is a part (e.g., platform).
- **Project Name–**The name of the software project.
- **Version**–The number and name of the version or release being described.
- **Process Description**–A comprehensive description of the standard software maintenance process being followed, preferably in an existing external document (e.g., Software Development Plan)
- **Application Domain–**Identify the domain as one of the following: avionics, business, command & control, microcode, process control, real-time, scientific, systems software, and telecommunications.
- **Platform**–The platform type of the system of which the software is a part: manned aircraft, unmanned aerial vehicle (UAV), ground fixed, ground mobile, unmanned space, missiles, or shipboard.

Sizing–Source Lines of Code (Mandatory)

The size of the software counted in non-blank, non-comment logical source lines of code (SLOC). Counting conventions for logical source lines vary by language. However, counters exist and should be used to count source lines for the language in question using conventions established by the Software Engineering Institute (SEI) in the following referenced standard:

- Robert E. Park, ***Software Size Measurement: A Framework for Counting Source Statements***, Technical Report CMU/SEI-92-TR-020, 1992.

The preferred code counter is the aforementioned USC Unified Code Counting (UCC) tool, which can be downloaded free from http://sunset.usc.edu.

If other measures of size, such as function points or object points, are used in addition to or in lieu of SLOC, they should be reported as well.

This set of data is being collected to define the size of the release, which is generally thought to be a driver of software effort. The data to be reported in this category includes:

- **Programming language(s)–**The programming language(s) in which the software version or release was written (including assembly).
- **New (added)–**The number of new human-generated SLOC added to the new version or release.
- **Auto-generated**–The number of auto-generated SLOC added to the new version or release. Auto-generated code is produced using specialized tools at a pace far exceeding manual development.
- **Carryover (existing)**–The number of SLOC from the previous version that were carried over as is. These lines are not changed in any way.
- **Reused (internal)**–The number of existing SLOC from a different project within the organization that were included in the new version or release. These lines are not changed in any way.

- **Reused (external)**–The number of existing SLOC from a different project outside the organization (e.g., Open Source) that were included in the new version or release. These lines are not changed in any way.
- **Modified (changed)**–The number of existing SLOC that were changed and included in the new version or release. These lines can include design modified, code modified and/or integration modified elements. Please specify source of modified code (previous release, internal, external) and degree of modification.
- **Deleted–**The number of existing SLOC that were deleted from the previous version or release.

Schedule (Mandatory)

The schedule represents the calendar time spent to generate the version or release from its start to its actual delivery date. This set of data is being collected to enable the prediction of schedule and to relate effort and staffing. The software effort starts when allocated software requirements are provided to the software team by systems engineering. The software effort ends when the Formal Qualification Tested (FQT'd) software is delivered to systems engineering for integration and test, typically in some System Integration Lab or facility. Schedule should be reported with interim milestones where tracked. (A possible set of milestones is Software Requirements, Preliminary Design, Detailed Design, Code & Unit Test, and Software I&T). The data to be reported in this category includes:

- **Estimated Begin Date**–The estimated calendar date that work on the new version or release should have began.
- **Actual Begin Date**–The actual calendar date that work on the new version or release began. This may differ from the estimated date due to any number of reasons.
- **Estimated End Date**–The estimated calendar date that the new version or release should have been delivered to systems engineering for integration and test.
- **Actual End Date**–The actual calendar date that the new version or release was delivered to systems engineering for integration and test.

Effort (Mandatory)

The effort represents the number of staff-hours spent during the time from when allocated software requirements are provided to when the FQT'd software is delivered to systems engineering for integration and test. The number of hours includes all directly-chargeable hours to the software project, including all of those expended by management, development, test and support personnel involved in getting the software product delivered, and including sustaining engineering. Effort should be reported by activity where tracked. (A possible set of activities is Software Requirements, Preliminary Design, Detailed Design, Code & Unit Test, Software I&T, Qualification Testing, Software Program Management, Software Quality Assurance, Software Configuration Management, Information Assurance, and Independent Verification and Validation.) The data to be reported in this category includes:

- **Estimated Effort (staff-hours)–**The estimated effort in staff-hours for the new version or release provided prior to the work begins.
- **Actual Effort (staff-hours)**–The actual effort expended in staff-hours for the new version or release provided when the work was completed.

- **Standard Month**–The number of staff-hours in a standard staff-month (only required if effort is only available in staff-months).
- **Labor Mix**–The breakout of staff-hours by labor category (e.g., senior/mid/junior).
- **Staffing Level**–identify the average number of people on the maintenance team and the peak staff size expressed as average (peak) for each version or release. If the data are available, record the composition of the team (e.g., ten average; one manager, six software engineers, one CM/QA person, one network administrator/security, and one field support engineer).
- **Labor Rates**–The fully-burdened dollars per hour ($/hr), either composite or by labor category. Can refer to standard documentation (e.g., rate schedules).

Quality (Mandatory)

The number of defects is determined by the tallying the number of Software Problem Reports (SPR) as they are entered into the problem reporting system. A defect is an error, flaw, mistake or fault in a software program that causes it to produce either incorrect or unexpected results, or causes it to behave in untended ways. Defects are sometimes separately by phase in which they are discovered in an attempt to determine how many escape detection in-phase and out-of-phase. If there are change requests separate from SPRs and formal requirements (see below), please provide similar counts of those as well.

This set of data is being collected to define the relative quality of the release as a potential cost driver. The data to be reported in this category includes:

- **Number of Defects–**The actual number of defects related to this version or release separated into the following five categories:
  - **Category 1 Defects (Catastrophic)**–The number of catastrophic defects related to this release. Catastrophic defects are those that prevent the accomplishment of an operational or mission-essential capability and for which no work-around solution is known. In addition, catastrophic defects include all system/software lockups and those defects that jeopardize safety, security, or other absolutely essential requirements.
  - **Category 2 Defects (Critical)**–The number of critical defects related to this release. Critical defects are those that adversely affect the accomplishment of an operational or mission-essential capability and for which a work-around solution is not known. In addition, such defects include those that adversely affect technical, cost, or schedule risks to the project or to life cycle support of the system and for which no work-around solution is known.
  - **Category 3 Defects (Serious)**–The number of serious defects related to this release. Serious defects are those that adversely affect the accomplishment of an operational or mission-essential capability, but for which a work-around solution is known.
  - **Category 4 Defects (Annoyance)**–The number of annoyance defects related to this release. Annoyance defects are those that typically result in user/operator inconvenience, but do not affect any required operational or mission-essential capability.
  - **Category 5 Defects (Minimal)**–The number of defects that both have minimal impacts and do not appear in any other category related to this release. They may be provided for informational purposes.

- **Defect Information** - Information supplied for defects in each of these categories, via a spreadsheet or table, includes:
    - Number of known defects; i.e., those existing prior to this release
    - Number of known defects planned to be fixed as part of this release
    - Number of known defects actually fixed as part of this release
    - Number of new defects found during work on this release
    - Number of new defects fixed as part of this release

Capability (Mandatory)

This information captures the overall skill of the software team. The data to be reported in this category includes:

- **Process Maturity**–The Capability Maturity Model (CMM) rating provided by SEI.
- **Application Experience–**The average number of years of experience of the software team with developing and maintaining this type of application.
- **Platform Experience–**The average number of years of experience of the software team with developing and maintaining software for this type of platform.
- **Language/Tool Experience–**The average number of years of experience of the software team with developing and maintaining software coded in this language and using this suite of software tools.

Cost (Optional)

The cost represents the dollars ($) spent during the time from when allocated software requirements are provided to when the FQT'd software is delivered to systems engineering for integration and test. The number of dollars ($) differs from effort in staff-hours as it includes all those expended on the project including those spent on licenses, travel, and other costs. The data to be reported in this category includes:

- **Estimated Labor Costs ($)–**The estimated labor costs in $ for the new version or release prior to the work on it being started.
- **Actual Labor Costs ($)**–The actual labor costs expended in $ for the new version or release when the work on it was completed.
- **Estimated License Costs ($)**–The estimated license costs in $ for the new version or release prior to the work on the new version it being started.
- **Actual License Costs ($)**–The actual license costs expended in $ for the new version or release when the work on it was completed.
- **Estimated Travel Costs ($)**–The estimated travel costs in $ for the new version or release prior to the work on it being started.
- **Actual Travel Costs ($)**–The actual travel costs expended in $ for the new version or release when the work on it was completed.
- **Estimated Facility Costs ($)**–The estimated costs for software development and test facilities in $ needed to sustain, test, and support of the new version or release, prior to the work on it being started. Does not include building costs (e.g., lease).
- **Actual Facility Costs ($)**–The actual costs for software development and test facilities in $ needed to sustain, test, and support the new version or release, when the work on it was completed. Does not include building costs (e.g., lease).

- **Estimated Other Costs ($)**–The estimated other direct costs (ODCs), not including Travel, in $ for the new version or release prior to the work on it being started. Includes separate Security/IA costs.
- **Actual Other Costs ($)**–The actual other direct costs (ODCs), not including Travel, expended in $ for the new version or release when the work on it was completed. Includes separate Security/IA costs.

Requirements (Optional)

If the maintenance effort is driven by requirements, they should be elicited, defined at a detailed level, and managed using a tool such as DOORS by IBM/Rational. Requirements are expressed in a complete sentence containing both a subject and predicate. These sentences shall consistently use the verb "shall" or "will" or "must" to show the requirement's mandatory nature. The whole requirement specifies a desired end goal or result and contains success criterion or other measurable indication of quality.

This set of data is being collected to substantiate budgets for software enhancements including funds needed for sustaining engineering and product support during operations. The data to be reported in this category includes:

- **Added–**The number of new requirements added to the current version or release.
- **Deleted**–The number of existing requirements deleted from the previous version or release.
- **Changed**–The number of existing requirements modified for the current version or release.
- **Deferred**–The number of requirements deferred from the new version or release solely due to funding constraints.
- **Total # Requirements**–The actual number of requirements in the new version or release when it is delivered for operational use.

Earned Value (Optional)

Earned value is a project management technique used to measure progress in an objective manner. It combines measurement of scope, schedule and cost into an integrated framework for determining status and assessing progress. If EVM is being conducted for this project, the below elements should be reported at lowest level of the work breakdown structure (WBS) for which they are collected. The data to be reported in this category includes:

- **Budgeted Cost of Work Performed (BCWP)**–the budgeted cost of the work actually completed.
- **Actual Cost of Work Performed (ACWP)–**the actual cost of the work completed taken from the financial records.
- **Budgeted Cost of Work Scheduled (BCWS)**–the budgeted cost of the work scheduled but not performed as of yet.
- **Budget At Completion (BAC)**–the current budget allocated to complete the work.
- **Estimate At Completion (EAC)**–the current estimated cost to complete the work.

Test Effort (Optional)

The effort represents the number of staff-hours spent to perform Formal Qualification Test (FQT) on the software version or release. It does not include staff-hours for unit testing. However, it does include staff-hours needed to conduct dry runs and prepare automation scripts. The number of hours includes all directly-chargeable hours to the software project including all of those expended by management, test and support personnel involved in getting the software product delivered. Where available, the below quantities should be broken out by type of testing (e.g., Dry Run, Dry Run Regression, FQT, and FQT Regression). The data to be reported in this category includes:

- **Number of Test Cases–**The actual number of test cases developed for the new version or release separated into the following categories:
- **Test Case Effort (staff-hours)–**The actual effort expended in staff-hours for developing test cases for the new version or release separated into the following categories:
- **Number of Tests Run**–The actual number of tests run for the new version or release separated into the following categories:
- **Test Conduct Effort (staff-hours)**–The actual effort expended in staff-hours for conducting the testing of the new version or release separated into the following categories:
- **Test Cost ($)**–The actual test cost expended in $ for the new version or release separated into the following categories:

Model Information (Optional)

If the COCOMO II or SLIM cost model was used to prepare the estimates for cost, please provide a copy of the estimate file and basis for estimate for each version or release. Multiple files are needed, i.e., that containing the initial estimate and another that updates the drivers to reflect the estimated cost- and schedule-to complete at the end the fiscal year for multi-year projects and actuals at the end of the effort. As an example, the team may have planned to use experienced people for the job, but they may have had difficulties finding them because the technology involved was so antiquated. The result is that the initial estimate assumed applications experience (“APEX” for the COCOMO II cost model) was “High” when in actuality it was “Low” for the updates. The values for experience should be captured along with an explanation in each updated file (cost-to-complete and actual). If you do not have these files, please complete the following two tables.

The COCOMO II and SLIM models were selected because they represent packages for which our sponsor holds licenses. There are other software cost models that can fit the bill. We have elected not to capture data for them because of license issues. However, we encourage you to do so if you use some of these other models. Understanding the factors that impact the effort and duration estimates is extremely important because it gives you insight into the factors upon which cost varies.

1. Scale Factors

Rate the COCOMO II scale drivers. These are the factors in the exponent of the equation. When in doubt use the nominal setting. Please provide the two versions of this table that were requested.

	Very Low	Low	Nominal	High	Very High	Extra High	Estimate Rating
Precedentedness	Thoroughly un-precedented	Largely un-precedented	Somewhat un-precedented	Generally familiar	Largely familiar	Largely familiar	
Development Flexibility	Rigorous	Occasional relaxation	Some relaxation	General conformity	Some conformity	Some conformity	
Architecture/ Risk Resolution	Little 20%	Some 40%	Often 60%	Generally 75%	Mostly 90%	Mostly 90%	
Team Cohesion	Strongly adversarial	Occasionally cooperative	Moderately cooperative	Largely cooperative	Highly cooperative	Highly cooperative	
Process Maturity	CMM Level 1 (lower half)	CMM Level 1 (upper half)	CMM Level 2	CMM Level 3	CMM Level 4	CMM Level 5	

2. **Cost Drivers**

Rate the COCOMO II cost drivers. These factors are multiplied together to adjust the project cost to factors that have been found to influence over it. When in doubt use the nominal setting. Please provide the two versions of this table that were requested.

	Very Low	Low	Nominal	High	Very High	Extra High	Estimate Rating
Required Software Reliability	Slight in-convenience	Low, easily recoverable losses	Moderate, easily recoverable losses	High financial loss	Risk to human life		
Data Base Size		D/P < 10	10 $\leq$ D/P < 100	100 $\leq$ D/P < 1000	D/P $\geq$ 1000		
Product Complexity	Simple	Straight-forward	Routine, some math, multi-file	Processing intense	Interrupt-driven	Complex real-time	
Required Reusability		None	Across project	Across Program	Across Product Line	Across Multiple Product Lines	
Documentation Match to Life Cycle Needs	Many life cycle needs uncovered	Some needs uncovered	Right-sized to life cycle needs	Excessive for life cycle needs	Very excessive for lifecycle needs		
Execution Time Constraints			$\geq$ 50% use of available exec. time	70% use	85% use	95% use	

Main Storage Constraints			$\geq$ 50% use of available storage	70% use	85% use	95% use	
Platform Volatility		Major - 12 months Minor - 1 month	Major - 6 months Minor – 2 weeks	Major - 2 months Minor - 1 week	Major - 2 weeks Minor – - 2 days		
Analyst Capability	15^{th}percentile	35^{th} percentile	55^{th}percentile	75^{th}percentile	90^{th} percentile		
Programmer Capability	15^{th}percentile	35^{th} percentile	55^{th}percentile	75^{th}percentile	90^{th} percentile		
Personnel Continuity	48%/year	24%/year	12%/year	6%/year	3%/year		
Application Experience	$\leq$ 2 months	6 months	1 year	3 years	6 years		
Platform Experience	$\leq$ 2 months	6 months	1 year	3 years	6 years		

Language/Tool Experience	≤ 2 months	6 months	1 year	3 years	6 years		
Use of Software Tools	Edit, code, debug	Simple front-end, backend CASE, little integration	Basic life cycle tools, moderate integration	Strong, mature tools, moderate integration	Strong, mature tools, well integrated with processes		
Site–Collocation	International	Multi-city and multi-company	Multi-city and multi-company	Same city or metro area	Same building or complex	Fully co-located	
Site–Communications	Some phone, mail	Individual phone, FAX	Narrow-band email	Wide-band electronic comm.	Wideband electronic comm., some video conf.	Inter-active multi-media	
Required Development Schedule	75%of nominal	85%of nominal	100% of nominal	130% of nominal	160%of nominal		
Multiply these factors to get the Effort Multiplier Factor (EMF)							

APPENDIX B

A. ISPAN CORRELATION ANALYSIS

1. ISPAN FY06 and FY07

Correlations

	FTE Maintenance	SLOC	Maintenance and Defects	CSCIs
FTE Maintenance	1.0000	0.8863	0.3660	-0.0202
SLOC	0.8863	1.0000	0.6661	0.0096
Maintenance and Defects	0.3660	0.6661	1.0000	-0.1899
CSCIs	-0.0202	0.0096	-0.1899	1.0000

Multivariate Correlations Report for FY06 ISPAN Data

Correlations

	FTE Maintenance	SLOC	Maintenance and Defects	CSCIs
FTE Maintenance	1.0000	0.8733	0.8511	-0.4925
SLOC	0.8733	1.0000	0.9574	-0.2364
Maintenance and Defects	0.8511	0.9574	1.0000	-0.1186
CSCIs	-0.4925	-0.2364	-0.1186	1.0000

Multivariate Correlations Report for FY06 ISPAN Data Minus One Subprogram With a Singular CSCI

Correlations

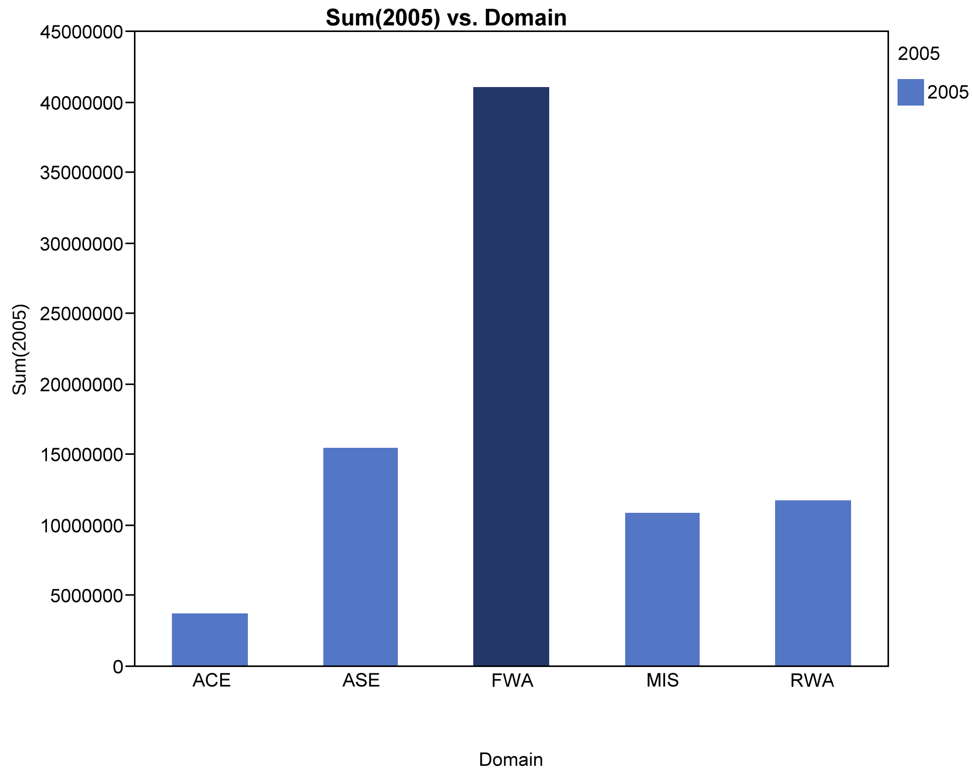
	FTE Maintenance	SLOC	Maintenance and Defects	CSCIs
FTE Maintenance	1.0000	0.7501	0.6454	0.0296
SLOC	0.7501	1.0000	0.8415	0.0170
Maintenance and Defects	0.6454	0.8415	1.0000	-0.1136
CSCIs	0.0296	0.0170	-0.1136	1.0000

Multivariate Correlations Report for FY07 ISPAN Data

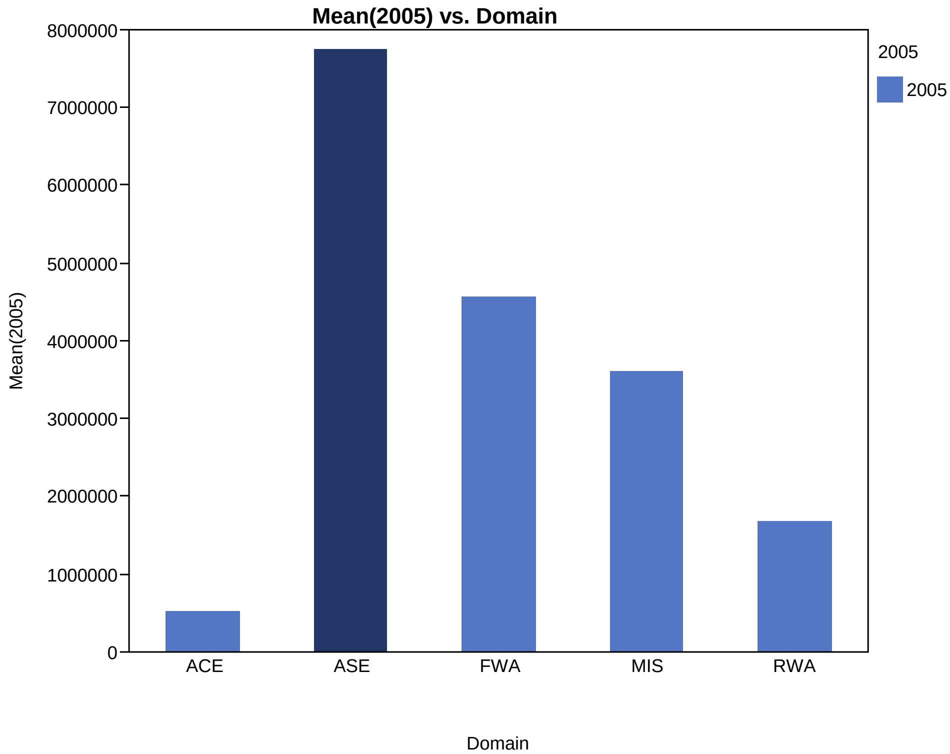
Correlations	FTE Maintenance	SLOC	Maintenance and Defects	CSCIs
FTE Maintenance	1.0000	0.6483	0.6218	-0.4660
SLOC	0.6483	1.0000	0.8173	-0.2185
Maintenance and Defects	0.6218	0.8173	1.0000	-0.2889
CSCIs	-0.4660	-0.2185	-0.2889	1.0000

Multivariate Correlations Report for FY07 ISPAN Data Minus One Subprogram With a Singular CSCI

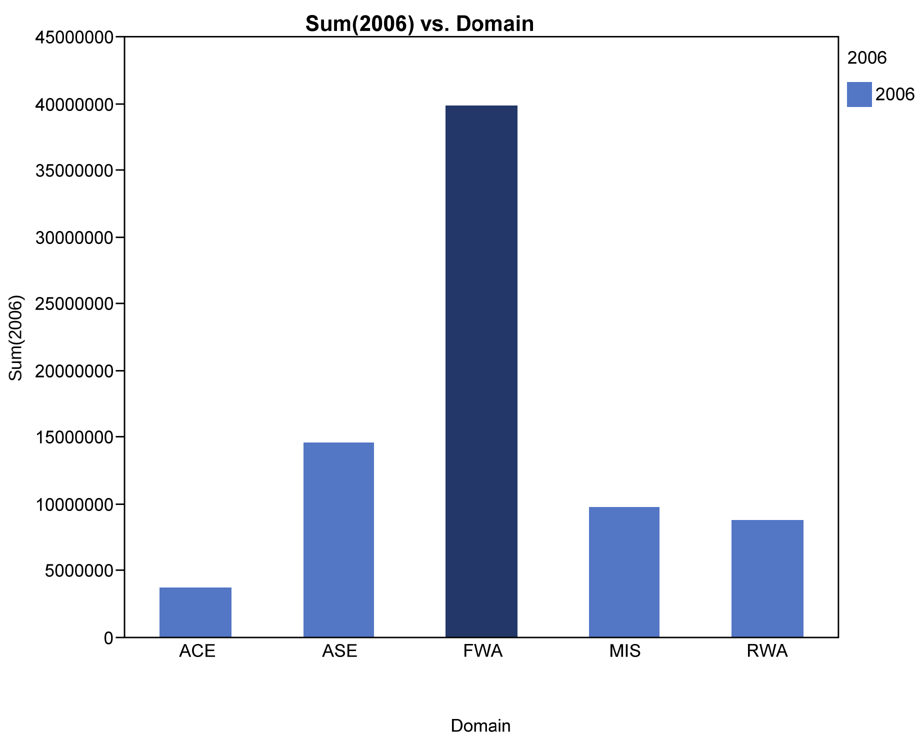
B. NAVAIR PRE DATA BY CATEGORY FOR FY05–FY07



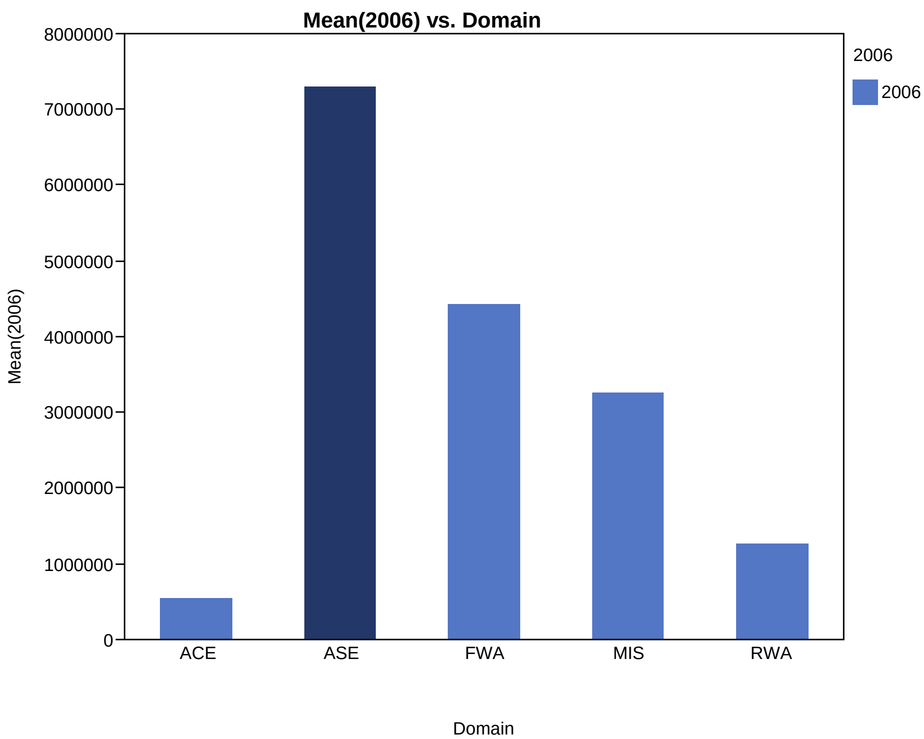
Sum of PRE Actual Funded Amount for FY05 by Category



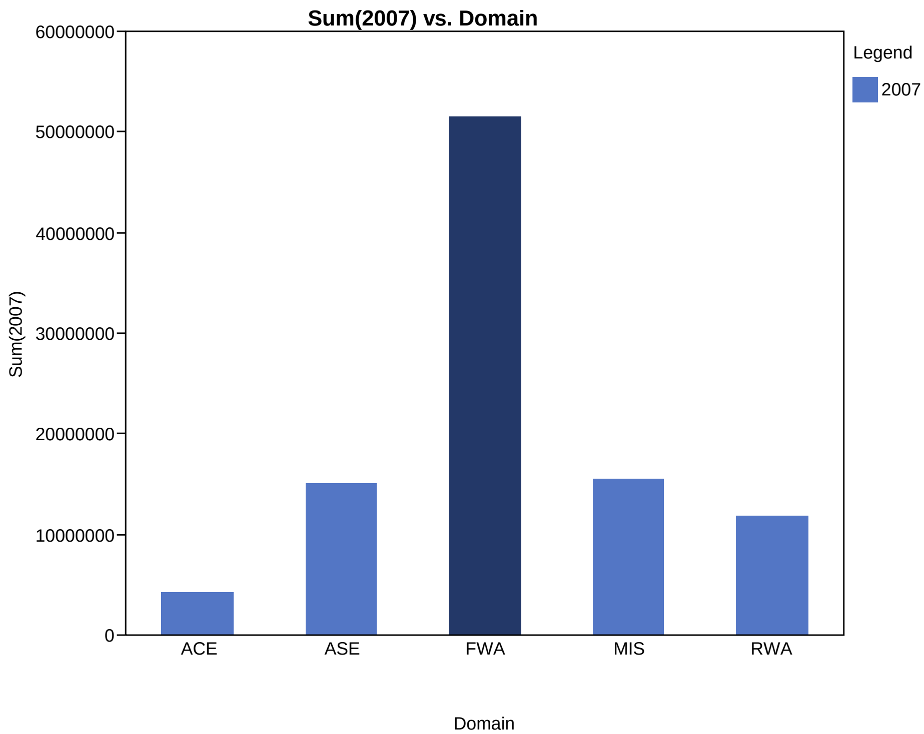
Mean of PRE Actual Amount Funded Amount for FY05 by Category



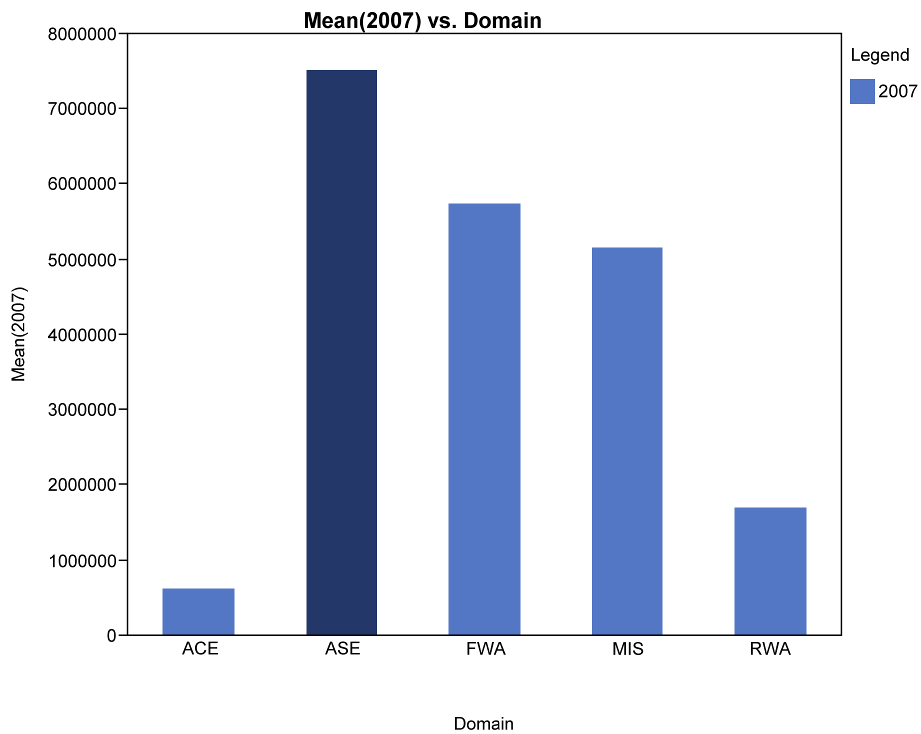
Sum of PRE Actual Amount Funded for FY06 by Category



Mean of PRE Actual Funded Amount for FY06 by Category



Sum of PRE Actual Funded Amount for FY07 by Category



Mean of PRE Actual Funding Amount for FY07 by Category

C. NAVAIR PRE CORRELATION ANALYSIS FOR FY04–FY07

1. Fixed Wing Aviation

Correlations

	FY04 Funded Amount	Avg of Units/Systems Deployed	SUM of SLOC	Sum of CSCI/Subsystems
FY04 Funded Amount	1.0000	0.6504	0.8783	0.9088
Avg of Units/Systems Deployed	0.6504	1.0000	0.6985	0.6566
SUM of SLOC	0.8783	0.6985	1.0000	0.7496
Sum of CSCI/Subsystems	0.9088	0.6566	0.7496	1.0000

Multivariate Correlations Report for PRE Data for Fixed Wing Aviation, FY04 Funded Amount, Average Number of Systems Deployed, SLOC, and CSCIs

Correlations

	FY05 Funded Amount	Avg of Units/Systems Deployed	SUM of SLOC	Sum of CSCI/Subsystems
FY05 Funded Amount	1.0000	0.5676	0.7743	0.8903
Avg of Units/Systems Deployed	0.5676	1.0000	0.6985	0.6566
SUM of SLOC	0.7743	0.6985	1.0000	0.7496
Sum of CSCI/Subsystems	0.8903	0.6566	0.7496	1.0000

Multivariate Correlations Report for PRE Data for Fixed Wing Aviation, FY05 Funded Amount, Average Number of Systems Deployed, SLOC, and CSCIs

Correlations

	FY06 Funded Amount	Avg of Units/Systems Deployed	SUM of SLOC	Sum of CSCI/Subsystems
FY06 Funded Amount	1.0000	0.6306	0.7922	0.8936
Avg of Units/Systems Deployed	0.6306	1.0000	0.6985	0.6566
SUM of SLOC	0.7922	0.6985	1.0000	0.7496
Sum of CSCI/Subsystems	0.8936	0.6566	0.7496	1.0000

Multivariate Correlations Report for PRE Data for Fixed Wing Aviation, FY06 Funded Amount, Average Number of Systems Deployed, SLOC, and CSCIs

Correlations

	FY07 Funded Amount	Avg of Units/Systems Deployed	SUM of SLOC	Sum of CSCI/Subsystems
FY07 Funded Amount	1.0000	0.9232	0.8502	0.8463
Avg of Units/Systems Deployed	0.9232	1.0000	0.6985	0.6566
SUM of SLOC	0.8502	0.6985	1.0000	0.7496
Sum of CSCI/Subsystems	0.8463	0.6566	0.7496	1.0000

Multivariate Correlations Report for PRE Data for Fixed Wing Aviation, FY07 Funded Amount, Average Number of Systems Deployed, SLOC, and CSCIs

2. Rotary Wing Aviation

Correlations

	FY04 Funded Amount	Avg of Units/Systems Deployed	Total SLOC	Total CSCIs/Subsystems
FY04 Funded Amount	1.0000	0.0279	0.6617	0.8941
Avg of Units/Systems Deployed	0.0279	1.0000	-0.3781	-0.2665
Total SLOC	0.6617	-0.3781	1.0000	0.9169
Total CSCIs/Subsystems	0.8941	-0.2665	0.9169	1.0000

Multivariate Correlations Report for PRE Data For Rotary Wing Aviation, FY04 Funded Amount, Average Number of Systems Deployed, SLOC, and CSCIs

Correlations

	FY05 Funded Amount	Avg of Units/Systems Deployed	Total SLOC	Total CSCIs/Subsystems
FY05 Funded Amount	1.0000	0.2500	0.4542	0.7272
Avg of Units/Systems Deployed	0.2500	1.0000	-0.3781	-0.2665
Total SLOC	0.4542	-0.3781	1.0000	0.9169
Total CSCIs/Subsystems	0.7272	-0.2665	0.9169	1.0000

Multivariate Correlations Report for PRE Data for Rotary Wing Aviation, FY05 Funded Amount, Average Number of Systems Deployed, SLOC, and CSCIs

Correlations

	FY06 funded Amount	Avg of Units/Systems Deployed	Total SLOC	Total CSCIs/Subsystems
FY06 funded Amount	1.0000	-0.3511	0.7645	0.8940
Avg of Units/Systems Deployed	-0.3511	1.0000	-0.3781	-0.2665
Total SLOC	0.7645	-0.3781	1.0000	0.9169
Total CSCIs/Subsystems	0.8940	-0.2665	0.9169	1.0000

Multivariate Correlations Report for PRE Data for Rotary Wing Aviation, FY06 Funded Amount, Average Number of Systems Deployed, SLOC, and CSCIs

Correlations

	FY07 Funded Amount	Avg of Units/Systems Deployed	Total SLOC	Total CSCIs/Subsystems
FY07 Funded Amount	1.0000	0.1404	0.5694	0.8366
Avg of Units/Systems Deployed	0.1404	1.0000	-0.3781	-0.2665
Total SLOC	0.5694	-0.3781	1.0000	0.9169
Total CSCIs/Subsystems	0.8366	-0.2665	0.9169	1.0000

Multivariate Correlations Report for PRE Data for Rotary Wing Aviation, FY07 Funded Amount, Average Number of Systems Deployed, SLOC, and CSCIs

3. Air Combat Electronics

Correlations

	FY04 Funded Amount	Avg of Units/Subsystems deployed	Total SLOC	Total CSCIs/Subsystems
FY04 Funded Amount	1.0000	0.7117	0.8545	0.7592
Avg of Units/Subsystems deployed	0.7117	1.0000	0.9449	0.7753
Total SLOC	0.8545	0.9449	1.0000	0.9125
Total CSCIs/Subsystems	0.7592	0.7753	0.9125	1.0000

Multivariate Correlations Report for PRE Data for Air Combat Electronic, FY04 Funded Amount, Average Number of Systems Deployed, SLOC, and CSCIs

Correlations

	FY05 Funded Amount	Avg of Units/Subsystems deployed	Total SLOC	Total CSCIs/Subsystems
FY05 Funded Amount	1.0000	0.1690	0.2595	0.0989
Avg of Units/Subsystems deployed	0.1690	1.0000	0.9449	0.7753
Total SLOC	0.2595	0.9449	1.0000	0.9125
Total CSCIs/Subsystems	0.0989	0.7753	0.9125	1.0000

Multivariate Correlations Report for PRE Data for Air Combat Electronic, FY05 Funded Amount, Average Number of Systems Deployed, SLOC, and CSCIs

Correlations

	FY06 Funded Amount	Avg of Units/Subsystems deployed	Total SLOC	Total CSCIs/Subsystems
FY06 Funded Amount	1.0000	0.2650	0.3767	0.2879
Avg of Units/Subsystems deployed	0.2650	1.0000	0.9449	0.7753
Total SLOC	0.3767	0.9449	1.0000	0.9125
Total CSCIs/Subsystems	0.2879	0.7753	0.9125	1.0000

Multivariate Correlations Report for PRE Data for Air Combat Electronic, FY06 Funded Amount, Average Number of Systems Deployed, SLOC and CSCIs

Correlations

	FY07 Funded Amount	Avg of Units/Subsystems deployed	Total SLOC	Total CSCIs/Subsystems
FY07 Funded Amount	1.0000	0.5626	0.6407	0.5131
Avg of Units/Subsystems deployed	0.5626	1.0000	0.9449	0.7753
Total SLOC	0.6407	0.9449	1.0000	0.9125
Total CSCIs/Subsystems	0.5131	0.7753	0.9125	1.0000

Multivariate Correlations Report for PRE Data for Air Combat Electronic, FY07 Funded Amount, Average Number of Systems Deployed, SLOC, and CSCIs

4. Air Combat Electronics and Aviation Support Equipment

Correlations

	FY04 Funded Amount	Avg of Units/Subsystems deployed	Total SLOC	Total CSCIs/Subsystems
FY04 Funded Amount	1.0000	-0.0818	-0.0079	0.7972
Avg of Units/Subsystems deployed	-0.0818	1.0000	0.4588	0.3980
Total SLOC	-0.0079	0.4588	1.0000	0.2895
Total CSCIs/Subsystems	0.7972	0.3980	0.2895	1.0000

Multivariate Correlations Report for PRE Data for Air Combat Electronic, FY04 Funded Amount, Average Number of Systems Deployed, SLOC, and CSCIs With ASE Data

Correlations

	FY05 Funded Amount	Avg of Units/Subsystems deployed	Total SLOC	Total CSCIs/Subsystems
FY05 Funded Amount	1.0000	-0.1096	-0.0234	0.7760
Avg of Units/Subsystems deployed	-0.1096	1.0000	0.4588	0.3980
Total SLOC	-0.0234	0.4588	1.0000	0.2895
Total CSCIs/Subsystems	0.7760	0.3980	0.2895	1.0000

Multivariate Correlations Report for PRE Data for Air Combat Electronic, FY05 Funded Amount, Average Number of Systems Deployed, SLOC, and CSCIs With ASE Data

Correlations

	FY06 Funded Amount	Avg of Units/Subsystems deployed	Total SLOC	Total CSCIs/Subsystems
FY06 Funded Amount	1.0000	-0.1075	-0.0205	0.7804
Avg of Units/Subsystems deployed	-0.1075	1.0000	0.4588	0.3980
Total SLOC	-0.0205	0.4588	1.0000	0.2895
Total CSCIs/Subsystems	0.7804	0.3980	0.2895	1.0000

Multivariate Correlations Report for PRE Data for Air Combat Electronic, FY06 Funded Amount, Average Number of Systems Deployed, SLOC, and CSCIs With ASE Data

Correlations

	FY07 Funded Amount	Avg of Units/Subsystems deployed	Total SLOC	Total CSCIs/Subsystems
FY07 Funded Amount	1.0000	-0.0794	0.0113	0.7945
Avg of Units/Subsystems deployed	-0.0794	1.0000	0.4588	0.3980
Total SLOC	0.0113	0.4588	1.0000	0.2895
Total CSCIs/Subsystems	0.7945	0.3980	0.2895	1.0000

Multivariate Correlations Report for PRE Data for Air Combat Electronic, FY07 Funded Amount, Average Number of Systems Deployed, SLOC, and CSCIs With ASE Data

5. Missiles

Correlations

	FY04 Funded Amount	Avg of Units/Systems Deployed	Total SLOC	Total CSCIs/Subsystems
FY04 Funded Amount	1.0000	0.9427	0.7139	0.1541
Avg of Units/Systems Deployed	0.9427	1.0000	0.4392	-0.1844
Total SLOC	0.7139	0.4392	1.0000	0.8020
Total CSCIs/Subsystems	0.1541	-0.1844	0.8020	1.0000

Multivariate Correlations Report for PRE Data for Missiles, FY04 Funded Amount, Average Number of Systems Deployed, SLOC, and CSCIs

Correlations

	FY05 Funded Amount	Avg of Units/Systems Deployed	Total SLOC	Total CSCIs/Subsystems
FY05 Funded Amount	1.0000	0.8453	-0.1087	-0.6810
Avg of Units/Systems Deployed	0.8453	1.0000	0.4392	-0.1844
Total SLOC	-0.1087	0.4392	1.0000	0.8020
Total CSCIs/Subsystems	-0.6810	-0.1844	0.8020	1.0000

Multivariate Correlations Report for PRE Data for Missiles, FY04 Funded Amount, Average Number of Systems Deployed, SLOC, and CSCIs

Correlations

	FY06 Funded Amount	Avg of Units/Systems Deployed	Total SLOC	Total CSCIs/Subsystems
FY06 Funded Amount	1.0000	0.2156	-0.7825	-0.9995
Avg of Units/Systems Deployed	0.2156	1.0000	0.4392	-0.1844
Total SLOC	-0.7825	0.4392	1.0000	0.8020
Total CSCIs/Subsystems	-0.9995	-0.1844	0.8020	1.0000

Multivariate Correlations Report for PRE Data for Missiles, FY06 Funded Amount, Average Number of Systems Deployed, SLOC, and CSCIs

Correlations				
	FY07 Funded Amount	Avg of Units/Systems Deployed	Total SLOC	Total CSCIs/Subsystems
FY07 Funded Amount	1.0000	0.4519	-0.6029	-0.9601
Avg of Units/Systems Deployed	0.4519	1.0000	0.4392	-0.1844
Total SLOC	-0.6029	0.4392	1.0000	0.8020
Total CSCIs/Subsystems	-0.9601	-0.1844	0.8020	1.0000

Multivariate Correlations Report for PRE Data for Missiles, FY07 Funded Amount, Average Number of Systems Deployed, SLOC, and CSCIs

6. Combination of Fixed and Rotary Wing Aviation

Correlations				
	FY04 Funded Amount	Avg of Units/Systems Deployed	Total SLOC	Total CSCI/Subsystems
FY04 Funded Amount	1.0000	0.4121	0.8909	0.8648
Avg of Units/Systems Deployed	0.4121	1.0000	0.4274	0.3395
Total SLOC	0.8909	0.4274	1.0000	0.7424
Total CSCI/Subsystems	0.8648	0.3395	0.7424	1.0000

Multivariate Correlations Report for PRE Data for Fixed and Rotary Wing Aviation, FY04 Funded Amount, Average Number of Systems Deployed, SLOC, and CSCIs

Correlations				
	FY05 Funded Amount	Avg of Units/Systems Deployed	Total SLOC	Total CSCI/Subsystems
FY05 Funded Amount	1.0000	0.3917	0.7875	0.8471
Avg of Units/Systems Deployed	0.3917	1.0000	0.4274	0.3395
Total SLOC	0.7875	0.4274	1.0000	0.7424
Total CSCI/Subsystems	0.8471	0.3395	0.7424	1.0000

Multivariate Correlations Report for PRE Data for Fixed and Rotary Wing Aviation, FY05 Funded Amount, Average Number of Systems Deployed, SLOC, and CSCIs

Correlations				
	FY06 Funded Amount	Avg of Units/Systems Deployed	Total SLOC	Total CSCI/Subsystems
FY06 Funded Amount	1.0000	0.3483	0.8282	0.8619
Avg of Units/Systems Deployed	0.3483	1.0000	0.4274	0.3395
Total SLOC	0.8282	0.4274	1.0000	0.7424
Total CSCI/Subsystems	0.8619	0.3395	0.7424	1.0000

Multivariate Correlations Report for PRE Data for Fixed and Rotary Wing Aviation, FY06 Funded Amount, Average Number of Systems Deployed, SLOC, and CSCIs

Correlations

	FY07 Funded Amount	Avg of Units/Systems Deployed	Total SLOC	Total CSCI/Subsystems
FY07 Funded Amount	1.0000	0.6353	0.8669	0.8096
Avg of Units/Systems Deployed	0.6353	1.0000	0.4274	0.3395
Total SLOC	0.8669	0.4274	1.0000	0.7424
Total CSCI/Subsystems	0.8096	0.3395	0.7424	1.0000

Multivariate Correlations Report for PRE Data for Fixed and Rotary Wing Aviation, FY07 Funded Amount, Average Number of Systems Deployed, SLOC, and CSCIs

LIST OF REFERENCES

Abran, A., & April, A. (2008). *Software maintenance management.* Hoboken, NJ: IEEE Computer Society.

Boehm, B., Abts, C., Brown, A., Chulani, S., Clark, B., Horowitz E., … & Steece, B. (2000). *Software cost estimation with COCOMO II.* Upper Saddle River, NJ: Prentice Hall.

Brooks, F., Jr. (1995). *The mythical man-month: Essays on software engineering, 20th anniversary.* Crawfordsville, IN: Addison Wesley Longman.

Buchmann, I., Frischbier, S., & Putz, D. (2011). Towards an estimation model for software maintenance costs. *Proceedings of the 2011 15th IEEE European Conference on Software Maintenance and Reengineering, 1534–5351*(11), 313–316.

Christensen, M., & Thayer, R. (2001). *The project manager's guide to software engineering best practices.* Los Alamitos, CA: IEEE Computer Society.

Defense spending in a time of austerity. (2010, August 26). *The Economist, 396*(8697), 20–22.

Department of the Air Force. (2000). *Guidelines for successful acquisition and management of software intensive systems: Weapons systems, command and control systems, management information system* (Version 3.0). Hill AFB, UT: Software Technology Support Center.

Department of Defense (DoD). (2004). *Software resources data report (SRDR) manual.* Retrieved from http://dcarc.pae.osd.mil//Files/Policy/DOD50004M2.pdf

Department of Defense (DoD). (2008). *Major automated information system annual report: Integrated strategic planning and analysis network.* Retrieved from the Acquisition Community Connection at the Defense Acquisition University website: https://acc.dau.mil

Galorath Incorporated. (2001). *SEER–SEM software estimation, planning and project control: User's manual.* El Segundo, CA: Author.

Galorath Incorporated. (2011a). *SEER for IT (information technology): Estimating IT projects, IT project estimation software, IT project planning.* Retrieved from http://www.galorath.com/DirectContent/SEERforIT2.pdf

Galorath Incorporated. (2011b). *Software maintenance cost estimation.* Retrieved from http://www.galorath.com/index.php/software_maintenance_cost

Naval Air Systems Command (NAVAIR). (2010). *NAVAIR software logistics primer, version 2.0.* Retrieved from the Acquisition Community Connection at the Defense Acquisition University website: https://acc.dau.mil

Nussbaum, D. (2010, Summer). Cost estimation techniques [Coursework, Class OA4702, session 6.2]. Graduate School of Operational and Information Sciences, Naval Postgraduate School, Monterey, CA.

Parthasarathy, M. (2007). *Practical software estimation.* Upper Saddle River, NJ: Addison Wesley.

Quantitative Software Management (QSM). (2006). *User's guide to SLIM Estimate 6.3 for Windows.* McLean, VA: Quantitative Software Management.

Reifer, D., Allen, J., Fersch, B., Hitchings, B., Judy, J., & Rosa, W. (2010). *Software maintenance: Software cost estimation metrics* (Initial draft). Washington, DC: Air Force Cost Analysis Agency.

Rendon, R., & Snider, K. (Eds.). (2008). *Management of defense acquisition projects.* Reston, VA: American Institute of Flight.

Rosa, W., & Braxton, P., (2010, November). *Software maintenance data collection overview*. PowerPoint presentation given at Headquarters U.S. Air Force, Air Force Cost Analysis Agency, Washington, DC.

Salkind, N. (2004). *Statistics for people who (think they) hate statistics.* Thousand Oaks, CA: Sage Publications.

Sneed, H. M. (2004). A cost model for software maintenance & evolution. *Proceedings of the 20th IEEE International Conference on Software Maintenance, 1063–6773*(4), 264–273.

Under Secretary of Defense for Acquisition, Technology, and Logistics (USD[AT&L]). (2009, December). *Implementation of the Weapons Systems Acquisition Reform Act of 2009* (Directive-type memorandum [DTM] 09-027). Washington, DC: Author.

United States Strategic Command (USSTRATCOM). (2004). *ISPAN modernization cost analysis requirements description (CARD)*. Omaha, NE: USSTRATCOM.

Weapons Systems Acquisition Reform Act of 2009, 10 U.S. C. § 2334 (2009).

2003 - 2011 SPONSORED RESEARCH TOPICS

Acquisition Management

- Acquiring Combat Capability via Public-Private Partnerships (PPPs)
- BCA: Contractor vs. Organic Growth
- Defense Industry Consolidation
- EU-US Defense Industrial Relationships
- Knowledge Value Added (KVA) + Real Options (RO) Applied to Shipyard Planning Processes
- Managing the Services Supply Chain
- MOSA Contracting Implications
- Portfolio Optimization via KVA + RO
- Private Military Sector
- Software Requirements for OA
- Spiral Development
- Strategy for Defense Acquisition Research
- The Software, Hardware Asset Reuse Enterprise (SHARE) repository

Contract Management

- Commodity Sourcing Strategies
- Contracting Government Procurement Functions
- Contractors in 21st-century Combat Zone
- Joint Contingency Contracting
- Model for Optimizing Contingency Contracting, Planning and Execution
- Navy Contract Writing Guide
- Past Performance in Source Selection
- Strategic Contingency Contracting
- Transforming DoD Contract Closeout
- USAF Energy Savings Performance Contracts
- USAF IT Commodity Council
- USMC Contingency Contracting

Financial Management

- Acquisitions via Leasing: MPS case

- Budget Scoring
- Budgeting for Capabilities-based Planning
- Capital Budgeting for the DoD
- Energy Saving Contracts/DoD Mobile Assets
- Financing DoD Budget via PPPs
- Lessons from Private Sector Capital Budgeting for DoD Acquisition Budgeting Reform
- PPPs and Government Financing
- ROI of Information Warfare Systems
- Special Termination Liability in MDAPs
- Strategic Sourcing
- Transaction Cost Economics (TCE) to Improve Cost Estimates

Human Resources

- Indefinite Reenlistment
- Individual Augmentation
- Learning Management Systems
- Moral Conduct Waivers and First-term Attrition
- Retention
- The Navy's Selective Reenlistment Bonus (SRB) Management System
- Tuition Assistance

Logistics Management

- Analysis of LAV Depot Maintenance
- Army LOG MOD
- ASDS Product Support Analysis
- Cold-chain Logistics
- Contractors Supporting Military Operations
- Diffusion/Variability on Vendor Performance Evaluation
- Evolutionary Acquisition
- Lean Six Sigma to Reduce Costs and Improve Readiness
- Naval Aviation Maintenance and Process Improvement (2)
- Optimizing CIWS Lifecycle Support (LCS)

- Outsourcing the Pearl Harbor MK-48 Intermediate Maintenance Activity
- Pallet Management System
- PBL (4)
- Privatization-NOSL/NAWCI
- RFID (6)
- Risk Analysis for Performance-based Logistics
- R-TOC AEGIS Microwave Power Tubes
- Sense-and-Respond Logistics Network
- Strategic Sourcing

Program Management

- Building Collaborative Capacity
- Business Process Reengineering (BPR) for LCS Mission Module Acquisition
- Collaborative IT Tools Leveraging Competence
- Contractor vs. Organic Support
- Knowledge, Responsibilities and Decision Rights in MDAPs
- KVA Applied to AEGIS and SSDS
- Managing the Service Supply Chain
- Measuring Uncertainty in Earned Value
- Organizational Modeling and Simulation
- Public-Private Partnership
- Terminating Your Own Program
- Utilizing Collaborative and Three-dimensional Imaging Technology

A complete listing and electronic copies of published research are available on our website: www.acquisitionresearch.org

THIS PAGE INTENTIONALLY LEFT BLANK

ACQUISITION RESEARCH PROGRAM
GRADUATE SCHOOL OF BUSINESS & PUBLIC POLICY
NAVAL POSTGRADUATE SCHOOL
555 DYER ROAD, INGERSOLL HALL
MONTEREY, CALIFORNIA 93943

www.acquisitionresearch.org